Batch

Un batch peut se faire ne J2EE mais aussi peut se faire plus simplement:

Soit la classe produit:

```
import java.util.Date;


public class Product {
    private long id;
    private String name;
    private String description;
    private double price;
    private boolean published;
    private Date lastUpdate;
    // getters, setters and toString() omitted
}
```

Nous allons mettre en place le batch suivant

- Lire les données du fichier plat products.csv
- Filtrer les enregistrements commençant par #
- Mappe chaque enregistrement CSV à une instance du produit POJO
- Valider les données du produit
- Traiter chaque enregistrement à l'aide de l'implémentation ProductProcessor

Le fichier CSV a filtrer est le suivant:

```
#id,name,description,price,published,lastUpdate
0001,product1,description1,2500,true,2014-01-01
000x,product2,description2,2400,true,2014-01-01
0003,,description3,2300,true,2014-01-01
0004,product4,description4,-2200,true,2014-01-01
0005,product5,description5,2100,true,2024-01-01
0006,product6,description6,2000,true,2014-01-01
```

Le code suivant permet de filtrer les données du fichier CSV et de les mettre dans un autre fichier:

```
import org.easybatch.core.filter.StartWithStringRecordFilter;
import org.easybatch.core.job.Job;
import org.easybatch.core.job.JobBuilder;
import org.easybatch.flatfile.DelimitedRecordMapper;
import org.easybatch.flatfile.FlatFileRecordReader;
import org.easybatch.validation.BeanValidationRecordValidator;

public class WithEasyBatch {

    public static void main(String[] args) throws Exception {
        org.easybatch.core.job.Job job = new JobBuilder()
                .reader(new
FlatFileRecordReader("D:\\MyJavaProjects\\etnic2\\lanterna\\test.csv"))
                .mapper(new DelimitedRecordMapper(Product.class, "id","name", "description",
"price","published", "lastUpdate"))
                .filter(new MyProductFilter())
                .processor(new ProductProcessor())
                .writer(new StandardOutputRecordWriter())
                .build();

        JobExecutor jobExecutor = new JobExecutor();
        JobReport report = jobExecutor.execute(job);
        jobExecutor.shutdown();

        System.out.println("job report = " + report);
    }

}
```

Avec comme processur métier:

```
public class ProductProcessor implements RecordProcessor<Record, Record> {

    public Record processRecord(final Record record) throws Exception {
        System.out.println("product = " + record.getPayload());
        return record;
    }
```

```
}
```

Et comme filtre

```
public class MyProductFilter implements RecordFilter<Record<Product>> {

	public Record<Product> processRecord(Record<Product> record) {
		
		// TODO Auto-generated method stub
		return record.getPayload().getPrice()>0?record:null;
	}

}
```

Revision #3
Created 18 November 2019 19:54:15 by Admin
Updated 18 November 2019 20:12:25 by Admin