SimpleCSV Package

Version 2.5
February 2019

Gray Watson

Table of Contents

SimpleCSV

Version 2.5 – February 2019

This package provides some Java classes to help with the reading and writing of CSV (Comma Separated Values) files.

To get started quickly using SimpleCSV, see Chapter 1 [Quick Start], page 2. You can also take a look at the examples section of the document which has various working code packages. See Chapter 3 [Examples], page 5. There is also a HTML version of this documentation.

Gray Watson http://256.com/gray/

1 Start Using Quickly

To use SimpleCSV you need to do the following steps. For more information, see Chapter 2 [Using], page 3.

1. Download SimpleCSV from the SimpleCSV release page. See Section 2.1 [Downloading], page 3.
2. Add `@CsvColumn` annotation to each of the fields or get/set method that you want to write and read to/from CSV files. See Section 2.3 [CsvColumn Annotation], page 3.

```
public class Account {
    ...
    @CsvColumn
    private String name;
```

   Or.

```
public class Account {
    ...
    @CsvColumn
    private String getName() {
        return name;
    }
    @CsvColumn
    private void setName(String name) {
        this.name = name;
    }
```

3. Create a CsvProcessor utility class for the entity.

```
CsvProcessor<Account> processor =
    new CsvProcessor<Account>(Account.class);
```

4. Write a collection of `Account` entities to disk in CSV format.

```
processor.writeAll(new File("accounts.csv"),
    accounts, true /* write header */);
```

5. Read in from a CSV file and get a collection of `Accounts`:

```
List<Account> accounts =
    processor.readAll(new File("accounts.csv"),
    true /* first line header */,
    true /* validate header */,
    null /* used to record parse errors */);
```

For more extensive instructions, see Chapter 2 [Using], page 3.

2 Using SimpleCSV

2.1 Downloading Jar

To get started with SimpleCSV, you will need to download the jar file. The SimpleCSV release page is the default repository but the jars are also available from the central maven repository.

The code works with Java 6 or later.

2.2 Using With Maven

To use SimpleCSV with maven, include the following dependency in your '`pom.xml`' file:

```
<dependency>
<groupId>com.j256.simplecsv</groupId>
<artifactId>simplecsv</artifactId>
<version>2.5</version>
</dependency>
```

2.3 CsvColumn Annotation

The `@CsvColumn` annotation is used to mark the fields in your entity that you want to write to and read from CSV files as a column. It also allows you to customize the output format and other details for the particular field instance. The following fields from the annotation can be used:

`columnName`
: This allows you to override and set a column name for the field. By default it will use the field name. This column name is used when you are generating and validating the header line.

`mustNotBeBlank`
: Set to true if a value in the column is required. This means that it cannot be empty when it is being read in and a parse error or exception will be generated.

`trimInput`
: Set to true if you want the column read from the line to be trimmed (using `String.trim()`) before it is converted to Java. This may not be applicable to all field types.

`format`
: Sets the format for this column. Not all types use the format specifier. Take a look at the particular converter class javadocs for more particulars. The default format tends to be the `toString()` of the type, and (for example) the `java.text.DecimalFormat` class is used to override for numbers.

`converterFlags`
: Optional flags for the converter which adjust the output. The flags that are used depend on the converter. See the converter Javadocs for more information. These need to be constants that are added together. For example,

```
@CsvColumn(converterFlags = XxxConverter.FLAG1 + XxxConverter.FLAG2)
private Xxx dollarAmount;
```

`converterClass`
: Sets the converter to use to convert this column if you don't want to use the default appropriate internal class. This will construct and instance of the class for this particular field. If you want to use a singleton then you should register the type using `CsvProcessor.registerConverter(...)`. This converter class must have a public no-arg constructor.

`defaultValue`
: Set this to a default string for the column. If the column is empty when read, the value will be used instead. Default is the empty string.

`mustBeSupplied`
: Set to false if a column is optional and can be skipped in the input altogether. If this is false then the column doesn't have to be in the header or the lines at all. Default is true.

  WARNING: If you are using optional ordering, the same CsvProcessor cannot be used with multiple files at the same time since the column lists can be dynamic depending on the input file being read.

`afterColumn`
: Used to set the order of the columns by setting the column-name that this column comes after. If this is not specified then the order in which the fields and methods are discovered in the classes will determine their order in the CSV file. If two fields say they come after the same field then you will get an undefined order. If there is an loop in the after columns then an exception will be thrown.

Here's some examples of how to use the `@CsvColumn` annotation.

Override the column name:

```
@CsvColumn(columnName = "Account Number")
private long number;
```

Change the column input/output format. This will display the amount as `$1,231.00` or `($2,000,000.28)`.

```
@CsvColumn(columnName = "Amount", format = "$###,##0.00;($###,##0.00)")
private double amount;
```

Specifying a custom converter class for an object that you have defined.

```
@CsvColumn(columnName = "Gender", converterClass = GenderConverter.class)
private Gender gender;
```

3 Example Code

Here is some example code to help you get going with SimpleCSV. I often find that code is the best documentation of how to get something working. Please feel free to suggest additional example packages for inclusion here. Source code submissions are welcome as long as you don't get piqued if we don't chose your's.

Simple, basic

This is a simple application which publishes a single object. See the source code on github.

4 Open Source License

Index of Concepts