Export

Export is one part of the Presentation Layer. Export "takes" the data from the database and stores them in a file. This raises the need to be able to handle several different file formats, each of which has its special requirements. Therefore the Export framework follows the Builder Design Pattern.

Introduction

The Export allows the User to select the list of records that are to be exported and specify which attributes of the Occurrences are important. The list of selected records is represented by a simple wrapper of a set called the **Selection -** the records are identified by their unique database identifier, there is no need to store the whole records. The list of selected attributes is represented by another wrapper of a set called the **Projection**. The Projection allows us to quickly determine, whether certain attribute of the record was selected or not.

Export also makes use of the last select query that was executed by the User in the Search. If the User selected the records from a smaller subset, it is convenient to use this subset as a frame for the Selection. The last search query is also required in order to achieve the proclaimed functionality: *nothing means everything* i.e. if the User didn't select any record, everyting from the last Search will be exported.

After all information needed to start the export are gathered, they are supplied to an Export Task Factory, that can create Export tasks (Directors of the Builder Pattern), and appropriate Builders according to the selected file format.

The Participants

There are several participants; they can be found in the net.sf.plantlore.client.export package.

Builder

The interface contains several methods that are used to build the output appropriately. The source is in the file net.sf.plantlore.client.export.Builder.java.

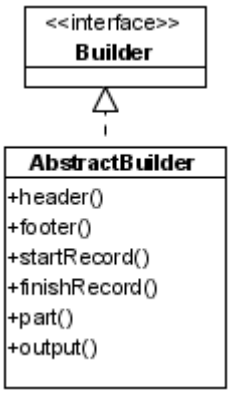
The Occurrence record is quite complicated and there are M:N relations in the database model. In order to process the Occurrence properly, it must be supplied to the Builder in parts. Those parts could be easily identified using the Java's reflection API, therefore it is not necessary to have several methods for each part of the record.

The Builder usually takes the Projections in order to know which attributes should be exported and which should not.

AbstractBuilder

AbstractBuilder adds (mostly empty) implementation to several methods of the interface and introduces new method that is used for the output. The advantage of the AbstractBuilder is that it decomposes the record into its attributes and uses the nethod **output()** to create the output. This way

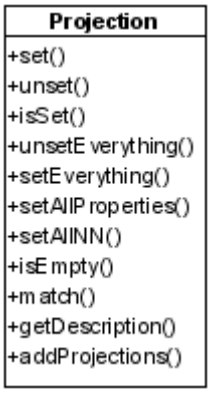
of implementing an operation is known as the Template Method Pattern.



Projection

The Projection is a simple wrapper of a java.util.Collection. It provides a very easy way of setting and unsetting a column and testing, whether a column was selected or not. It can also apply the selected list of columns as projection for the supplied query. To learn more about projections see the Section **Database Layer**. After the projections are added the caller should obtain the list containing the information which columns were projected in what order.

The source is in the file net.sf.plantlore.client.export.Projection.java.



Selection

The Selection stores the list of selected records and allows us to quickly determine whether a record was selected or not.

The source is in the file net.sf.plantlore.common.Selection.java.

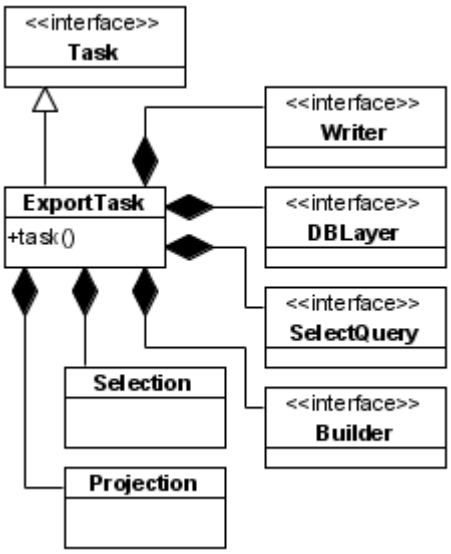
ExportTask

The Export task is the Director that manages the whole export. Export task requires the last Search query that will be used to iterate over, the Selection that contains the selected records from the

query (if any), and the Builder that constructs the output. It also requires Projection in case it should start using projections. The Writer is the interface for file operations - in this file the output will be stored.

The Export Task is implemented as a Task. To learn more about tasks see the Section **Tasks and Dispatcher**.

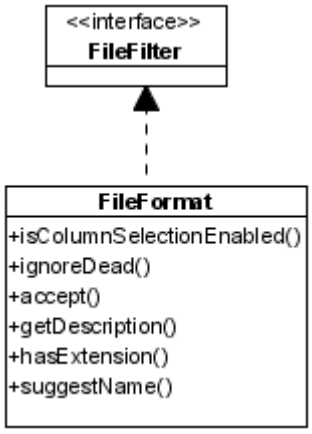
The Export Task is in the net.sf.plantlore.client.export.ExportTask.java file.



FileFormat

The File Format stores some information about a supported format of a file - whether all columns should be always exported, whether ignore records marked as dead, and the list of suitable extensions. The File Format implements File Filter which is a Java interface that must be implemented if programmer wants the filter to be used in the JFileChooser component.

The File Format is in the net.sf.plantlore.client.export.component.FileFormat.java file.

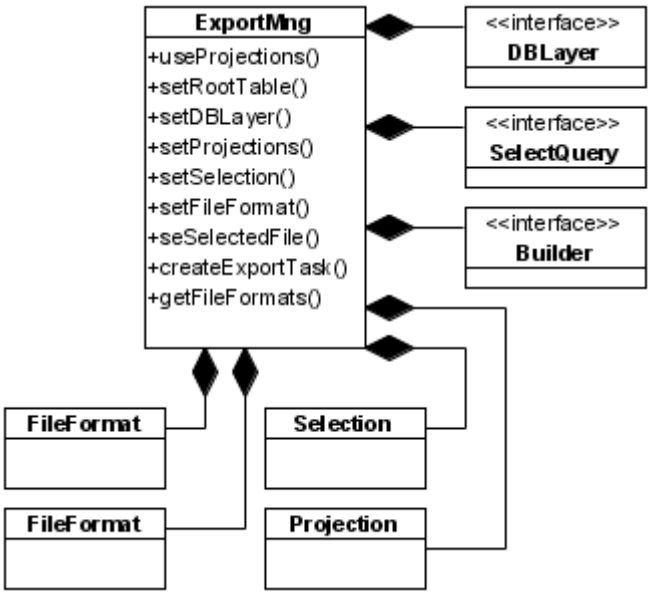


ExportManager

The Export Manager is the Export Task factory. It is used to store the partial information about the future Export task and once all the information required is supplied, it can create a new Builder, that will be used to form the final output, and the Export task that will manage the whole export.

The decision which Builder will be used is based on the supplied File Format. The Export Manager also stores a list of all available File Formats that can be handled by Builders known to the Export Manager.

The source is in the net.sf.plantlore.client.export.ExportMng.java file.



XMLBased Builders

All XMLbased builders use the DOM4J to construct parts of the output. It is not possible to use the DOM4J to build the whole output, because the XML tree is kept in memory and it could result in extreme memory requirements.

A combined solution was introduced. Only one record is constructed using the DOM4J at a time. When another record is received, the previous one is written to the output. This way it is possible to handle virtually any number of records and still use the greatest advantage of the DOM4J - XML tree construction for Occurrence records.

These Builders implement the Builder interface and are in the net.sf.plantlore.client.export.builders package.

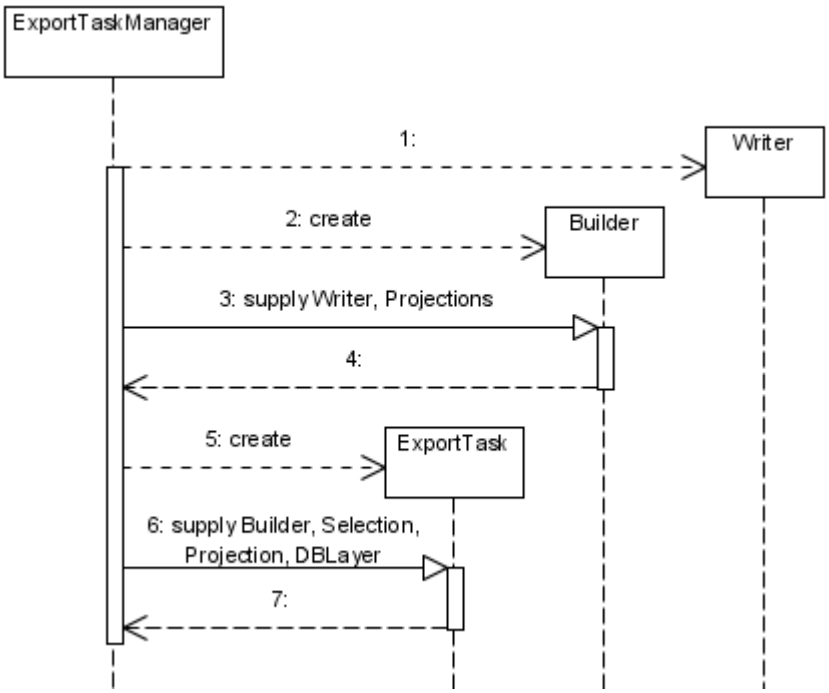
Comma Separated Values Builder

The CSV Builder places each Occurrence record on a separate line. If there is more than one Author associated with the Occurrence record, the Occurrence is repeated for every Author. That is: an Occurrence having *n* authors will be repeated three times, each time with a different Author.

This Builder extends the AbstractBuilder and can be found in the net.sf.plantlore.client.export.builders package.

Interaction

The creation of all participants



Start of the Export Task

The start of the Export task - the Header is created and then repeatedly action 5-15. are taken for every selected record.