Guide

Release 1.0.1

Author: Uwe Finke

http://cubaja.googlecode.com

config

Simple example

Imagine a batch application which reads records from a database and writes them into a file. The selection is restricted to records which were inserted within a certain time period. Database connection parameters, file name and time period have to be configurable in an XML file. We name it 'config.xml':

```
<?xml version="1.0"?>
<config dateFrom="2010-01-01"
        dateTo="2010-01-31">
  <database driver="com.mysql.jdbc.Driver"
            url="jdbc:mysql://localhost:3306/cubaja"
            user="cubaja"
            password="test"/>
  <output name="/path_to_file/output.txt"/>
</config>
```

A Java class corresponds to the structures of our XML; all attribute and element names match to setter methods by name. With an IDE, we only have to write the attribute types and names; the setter and getter methods are generated.

```
// imports ...
public class Config {
  private Date dateFrom;
  private Date dateTo;
  private DatabaseConfig database;
  private FileConfig output;
  public Config() {
  }
  public void setDateFrom(Date dateFrom) {
    this.dateFrom = dateFrom;
  }
  public void setDateTo(Date dateTo) {
    this.dateTo = dateTo;
  }
  public void setDatabase(DatabaseConfig database) {
    this.database = database;
  }
  public void setOutput(FileConfig output) {
    this.output = output;
  }
  public DatabaseConfig getDatabase() {
    return database;
  }
  // ... other getter methods ...
}
```

At runtime, we put the XML file's directory (which may be a common config directory) into the classpath. The application uses a Configurator to parse the XML and populate the Config object:

```
import de.ufinke.cubaja.config.Configurator;

// ...

Configurator configurator = new Configurator();
Config config = configurator.configure(new Config());
```

Protect the application against invalid data

Sometimes it may happen, e.g. because of a typo, that an XML configuration attribute or element doesn't match a setter method in the configuration class. Then the Configurator will notify us with a ConfigurationException.

The other way round, when there was no attribute or element in the config file which corresponds to an existing setter method, the Configurator doesn't complain about something missing. An attribute or element may be optional and a null or another default value may be used by the application. If there is no default value, we should protect our application against invalid data (which often leads to NullPointerExceptions). This is the purpose of the Mandatory annotation:

```
import de.ufinke.cubaja.config.Mandatory;

// ...

@Mandatory
public void setDatabase(DatabaseConfig database) {

  this.database = database;
}
```