BIDJI

User Manual

Bijdi

V0.5.0

Authors

Benjamin de Dardel, Bidji Project <bdedardel[_@t_]users.sourceforge.net>

Revision History

Revision V0.1	2012-12-01	Benjamin de Dardel
code generation with bidjic ant task		
Revision V0.2	2014-08-24	Benjamin de Dardel
- improve code generation		
- sql data transformation		
Revision V0.3	2014-11-11	Benjamin de Dardel
- text data transformation		
Revision V0.4	2015-08-30	Benjamin de Dardel
- csv data transformation		
- use cases (file2file, file2set, set2set)		
Revision V0.5	2017-07-10	Benjamin de Dardel
- xml data transformation		

Table of Contents

1. Transformation

1.1 Sql transformation

Description

Sqlt is an ant task that allows to:

- extract datas from database using ant sql task [https://ant.apache.org/manual/Tasks/sql.html]
- process datas using freemarker templates [http://freemarker.org/]

Sqlt task parameters

```
|Attribute  | Description           | Value                          | Required
|-----------|-----------------------|--------------------------------|---------
|template   | freemarker template   | <filename>.ftl                 | yes
|tofile     | output file           | <filename>.<mime>              | yes
|overwrite  | overwrite output file | true by default                | no
|Element    | Description           | Value                          | Required
|-----------|-----------------------|--------------------------------|---------
|sql        | ant sql task          |                                | yes
```

Options

Mapping

```
<mapping groupby="model_id">
<column index="0" as="parameter_name"/>
<column index="1" as="parameter_type"/>
<column index="2" as="parameter_type_id"/>
<column index="3" as="model_name"/>
<column index="4" as="model_id"/>
</mapping>
```

Examples

Extract bugs from a mantis [https://www.mantisbt.org/] bugtracker and write them into csv file.

build.xml

ant file:

- set *outputproperty*

```
<project name="bidji-sqlt-project" xmlns:bj="antlib:org.bidji.taskdefs">
<target name="mantis" description="generate mantis2csv">
     <bj:sqlt template="mantis2csv.ftl" tofile="mantis.csv" overwrite="true">
         <sql id="connection1"
            driver="com.mysql.jdbc.Driver"
            url="jdbc:mysql://$ {db.host}:3306/$ {db.name}"
            userid="readonly"
            password="readonly"
            print="true"
            encoding="UTF-8"
            delimiter="]["
            outputproperty="bugs"
            >
             <classpath>
              <pathelement path="lib/mysql-connector-java-5.1.22-bin.jar"/>
             </classpath>
         <![CDATA[
        SELECT b.id,b.category,b.fixed_in_version,b.summary,t.description,t.st
        FROM mantis_bug_table AS b
        INNER JOIN mantis_bug_text_table AS t ON t.id = b.bug_text_id
        ORDER BY b.id DESC;
        ]] >
         </sql>
     </bj:sqlt>
</target>
</project>
```

mantis2csv.ftl

- freemarker file

```
<#list bugs as bug>
$ {bug.id},$ {bug.summary},$ {bug.category},$ {bug.fixed_in_version}
</#list>
```

1.2 TXT TRANSFORMATION

Description

txtt is *text transformer* task that allow to:

- read datas from text file(s)
- process each line using freemarker templates [http://freemarker.org/]

Txtt task parameters

```
|Attribute  | Description           | Value                          | Required
|-----------|-----------------------|--------------------------------|---------
|file       | input file            | <filename>.txt                 | yes
|tofile     | output file           | <filename>.<mime>              | yes
|template   | freemarker template   | <filename>.ftl                 | yes
|overwrite  | overwrite output file | true by default                | no
```

Examples

Parse each line of a file and write a specific Java parser.

result.txt

- input file (result of command: *time my.bin*)

```
Command being timed: "./my.bin"
User time (seconds): 0.00
System time (seconds): 0.00
Percent of CPU this job got: 0%
Elapsed (wall clock) time (h:mm:ss or m:ss): 0:00.00
Average shared text size (kbytes): 0
Average unshared data size (kbytes): 0
Average stack size (kbytes): 0
Average total size (kbytes): 0
Maximum resident set size (kbytes): 1124
Average resident set size (kbytes): 0
Major (requiring I/O) page faults: 0
Minor (reclaiming a frame) page faults: 339
Voluntary context switches: 1
Involuntary context switches: 1
Swaps: 0
File system inputs: 0
```

```
File system outputs: 0
Socket messages sent: 0
Socket messages received: 0
Signals delivered: 0
Page size (bytes): 4096
Exit status: 0
```

build.xml

- ant file:

```
<project name="bidji-txtt" xmlns:bj="antlib:org.bidji.taskdefs">
<target name="txt2java" description=" transform txt to java">
    <bj:txtt file="result.txt" tofile="parser.java" template="txt2java.ftl" ov
</target>
</project>
```

txt2java.ftl

- freemarker file

```
// copy this function to your java code
public void parseLines(List<String> lines) {
    for (String line : lines) {
        switch (line.substring(0,line.indexOf(":")-1)) {
            <#foreach line in lines>
            case "${line?substring(0,line?index_of(':'))?replace('(','\\\\(')}
                // TODO
                break;
            </#foreach>
            default:
                throw new Exception("Fail to parse line " + line);
        }
    }
}
```

parser.java

- output file

```
// copy this function to your java code
public void parseLines(List<String> lines) {
    for (String line : lines) {
```

```
        switch (line.substring(0,line.indexOf(":")-1)) {
            case "Command being timed":
                // TODO
                break;
            case "User time \\(seconds)":
                // TODO
                break;
            case "System time \\(seconds)":
                // TODO
                break;
            case "Percent of CPU this job got":
                // TODO
                break;
            case "Elapsed \\(wall clock) time \\(h":
                // TODO
                break;
            case "Average shared text size \\(kbytes)":
                // TODO
                break;
            case "Average unshared data size \\(kbytes)":
                // TODO
                break;
            case "Average stack size \\(kbytes)":
                // TODO
                break;
            case "Average total size \\(kbytes)":
                // TODO
                break;
            case "Maximum resident set size \\(kbytes)":
                // TODO
                break;
            case "Average resident set size \\(kbytes)":
                // TODO
                break;
            case "Major \\(requiring I/O) page faults":
                // TODO
                break;
            case "Minor \\(reclaiming a frame) page faults":
                // TODO
                break;
            case "Voluntary context switches":
                // TODO
                break;
            case "Involuntary context switches":
                // TODO
```

```
                break;
            case "Swaps":
                // TODO
                break;
            case "File system inputs":
                // TODO
                break;
            case "File system outputs":
                // TODO
                break;
            case "Socket messages sent":
                // TODO
                break;
            case "Socket messages received":
                // TODO
                break;
            case "Signals delivered":
                // TODO
                break;
            case "Page size \\(bytes)":
                // TODO
                break;
            case "Exit status":
                // TODO
                break;
            default:
                throw new Exception("Fail to parse line " + line);
        }
    }
}
```

1.3 CSV TRANSFORMATION

Description

Csvt is *csv transformer* task that allow to:

- read datas from csv file(s)
- process datas using freemarker templates [http://freemarker.org/]

Csvt task parameters

```
|Attribute  | Description           | Value                          | Required
|-----------|-----------------------|--------------------------------|---------
```

```
|file       | input file            | <filename>.csv                | yes
|tofile     | output file           | <filename>.<mime>             | yes
|template   | freemarker template   | <filename>.ftl                | yes
|delimiter  | csv delimiter         | ; by default                  | no
|overwrite  | overwrite output file | true by default               | no
```

Example

Transform csv datas to xml.

input.csv

- input file

```
class;property;type
MyTodolist;id;integer
MyTodolist;label;string
MyTask;id;integer
MyTask;label;string
MyTask;listId;integer
```

build.xml

- ant file:

```
<project name="bidji-csvt" xmlns:bj="antlib:org.bidji.taskdefs">
<target name="csv2xml" description=" transform csv datas to xml">
    <bj:csvt file="input.csv" tofile="output.xml" template="csv2xml.ftl" delim
</target>
</project>
```

csv2xml.ftl

- freemarker file

```
<module>
[#assign currentClass='']
[#foreach line in lines]
  [#if currentClass != line.class]
    [#if currentClass != '']
    </class>
    [/#if]
    <class name="${line.class}">
```

```
    [#assign currentClass=line.class]
  [/#if]
        <property name="${line.property}" type="${line.type}">
  [#if !line?has_next]
    </class>
  [/#if]
[/#foreach]
</module>
```

output.xml

- result file

```
<module>
    <class name="MyTodolist">
        <property name="id" type="integer">
        <property name="label" type="string">
    </class>
    <class name="MyTask">
        <property name="id" type="integer">
        <property name="label" type="string">
        <property name="listId" type="integer">
    </class>
</module>
```

1.4 XML TRANSFORMATION

Description

Xmlt is *xml transformer* task that allow to:

- read datas from xml file(s)
- process datas using freemarker templates [http://freemarker.org/]

Indentation

see: http://freemarker.org/docs/dgui_misc_whitespace.html

Xml task parameters

```
|Attribute  | Description           | Value                          | Required
|-----------|-----------------------|--------------------------------|---------
|file       | input file            | <filename>.xml                 | yes
```

```
|tofile     | output file           | <filename>.<mime>              | yes
|template   | freemarker template   | <filename>.ftl                 | yes
|overwrite  | overwrite output file | true by default                | no
```

Exemple

Transform xml datas to sql.

users.xml

- input file

```
<?xml version="1.0"?>
<users>
    <user login="foo" email="foo@mail.com"/>
    <user login="baz" email="baz@mail.com"/>
</users>
```

build.xml

- ant file:

```
<project name="bidji-xmlt" xmlns:bj="antlib:org.bidji.taskdefs">
<target name="xml2sql" description=" transform xml datas to sql">
    <bj:xmlt file="users.xml" tofile="users.sql" template="xml2sql.ftl"  overw
</target>
</project>
```

xml2sql.ftl

- freemarker file

```
INSERT INTO table_users ( login, mail )
VALUES
[#foreach user in users.user]
( '${user.@login}', '${user.@email}' )[#if user?is_last];[#else],[/#if]
[/#foreach]
```

users.sql

- result file

```
INSERT INTO table_users ( login, mail )
```

```
VALUES
( 'foo', 'foo@mail.com' ),
( 'baz', 'baz@mail.com' );
```

2. Generation

2.1 Bidjic compiler

Description

Bidjic is a *bidji file* compiler. It can be used :

- as an ant task

Bidji is a syntax used to describe :

- Data models : Bidjim(odel) files

Bidji can be written in :

- xml (by default)

Bidjim

- Firstable, bidji models are used to define the business domain [http://en.wikipedia.org/wiki/Business_domain] and entity–relationship [http://en.wikipedia.org/wiki/Entity_class] in a Model Driven Development approach.
- bidji models are also used to define class diagrams [http://en.wikipedia.org/wiki/Class_diagram] for any modules/components/classes of the system.

Syntax

```
<?xml version="1.0"?>
<bidji>
  <entity name="Baz">
    <property name="id" type="Integer" />
    <property name="name" type="String" />
  </entity>
</bidji>
```

Bidjic task parameters

```
|Attribute  | Description           | Value                | Required      |
|-----------|-----------------------|----------------------|---------------|
|file       | inpput file           | <filename>.<bidji*> | yes
                                                            (if no fileset)
|tofile     | output file           | <filename>.<mime>    | yes           |
```

```
|todir      | output dir            | <dirname>           | no             |
                                                          (if no tofile)
|template   | freemarker template   | <filename>.ftl      | yes            |
|overwrite  | overwrite output file | true by default     | no             |
|-----------|-----------------------|---------------------|----------------|
|Element    | Description           | Value               | Required       |
|-----------|-----------------------|---------------------|----------------|
|fileset    | file set              |                     | no             |
|filelist   | file list             |                     | no             |
|templateset| template set          |                     | no             |
```

Use cases

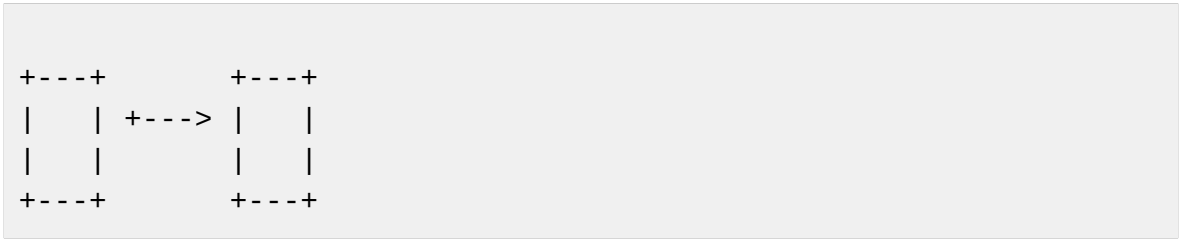
- File to file [bidjic-file-to-file.html]
- File to Set [bidjic-file-to-set.html]
- Set to Set [bidjic-set-to-set.html]

3. Use cases

3.1 Bidjic file to file

Description

Generates one output file from one input model.



Bidjic task parameters

Attribute	Description	Value	Required
file	inpput file	<filename>.<bidji*>	yes
tofile	output file	<filename>.<mime>	yes
overwrite	overwrite output file	true by default	no

Example

Generates Java POJO [https://en.wikipedia.org/wiki/Plain_Old_Java_Object] classes from a Model driven architecture [https://en.wikipedia.org/wiki/Model-driven_architecture] abstraction: the bidjim entity.

download [http://sourceforge.net/projects/bidji/files/samples/file2file.tgz/download] file2file.tgz sample.

ant file : build.xml

```
<target name="pojo" description="generate pojo classes">
  <bidjic file="MyPackage/MyModule/Foo.bidjim" tofile="gen/Foo.java" template=
</target>
```

template: bidjim2pojo.ftl

```
[#assign entity = bidji.entity]
package com.mycompany.${package}.${module};

/**
 * ${Model}, generated with bidjim2pojo.ftl
 **/
public class ${Model} {

[#foreach property in entity.property]
    private ${property.@type} _${property.@name};
[/#foreach]


   /**
    *  Constructor for a ${Model} Object.
    **/
    public ${Model}(){

    }
[#foreach property in entity.property]

    public ${property.@type?replace('Integer','int')} get${property.@name?cap_
        return this._${property.@name};
    }

    public void set${property.@name?cap_first}(${property.@type?replace('Integ
        this._${property.@name} = ${property.@name};
    }
[/#foreach]

}
```

model: MyPackage/MyModule/Foo.bidjim

```
<?xml version="1.0"?>
<bidji>
<entity name="Foo">
<property name="id" type="Integer" />
<property name="name" type="String" />
</entity>
</bidji>
```

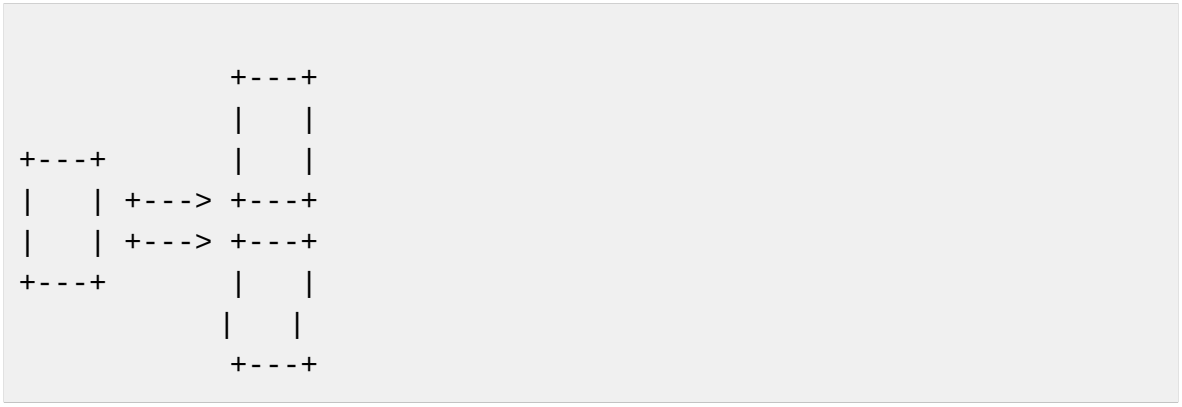
result: Foo.java

```
package com.mycompany.mypackage.mymodule;

/**
 * Foo, generated with bidjim2pojo.ftl
 **/
public class Foo {

    private int _id;
    private String _name;

   /**
    *  Constructor for a Foo Object.
    **/
    public Foo(){

    }

    public int getId() {
        return this._id;
    }

    public void setId(int id) {
        this._id = id;
    }

    public String getName() {
        return this._name;
    }

    public void setName(String name) {
        this._name = name;
    }

}
```

3.2 Bidjic file to set

Description

Generates multiple output files from one input model, using a templateset.



Bidjic task parameters

Attribute	Description	Value	Required
file	inpput file	<filename>.<bidji*>	yes
todir	output dir	<dirname>	yes
overwrite	overwrite output file	true by default	no

Element	Description	Value	Required
templateset	template set		yes

Example

Generates Php classes from from a Model driven architecture [https://en.wikipedia.org/wiki/Model-driven_architecture] abstraction: the bidjim entity.

download [http://sourceforge.net/projects/bidji/files/samples/file2set.tgz/download] file2set.tgz sample.

ant file : build.xml

```
<target name="php" description="generate php files">
    <bidjic file="MyPackage/MyModule/Foo.bidjim" todir="gen" overwrite="true">
        <templateset dir="templates">
            <include name="*.ftl"/>
        </templateset>
    </bidjic>
</target>
```

Note

Output file names are processed from template names:

- *Model*.php.ftl + MyPackage/MyModule/Foo.bidjim => *Foo*.php
- *Module*.php.ftl + MyPackage/MyModule/Foo.bidjim => *MyModule*.php

template: iModel.php.ftl

```
[#assign entity = bidji.entity]
<?php
namespace ${Package}\${Module};

interface i${Model}
{
[#foreach property in entity.property]
    public function get${property.@name?cap_first}();
    public function set${property.@name?cap_first}($${property.@name});
[/#foreach]
}
```

template: Model.php.ftl

```
[#assign entity = bidji.entity]
<?php
namespace ${Package}\${Module};

class ${Model} implements i${Model}
{
[#foreach property in entity.property]
    private $${property.@name};

    public function get${property.@name?cap_first}() {
        return $this->${property.@name};
    }

    public function set${property.@name?cap_first}($${property.@name}) {
        $this->${property.@name} = $${property.@name};
    }

[/#foreach]
}
```

model: MyPackage/MyModule/Foo.bidjim

```
<?xml version="1.0"?>
<bidji>
<entity name="Foo">
<property name="id" type="int" />
<property name="name" type="string" />
</entity>
</bidji>
```

result: iFoo.php

```
<?php
namespace MyPackage\MyModule;

interface iFoo
{
    public function getId();
    public function setId($id);
    public function getName();
    public function setName($name);
}
```

result: Foo.php

```
<?php
namespace MyPackage\MyModule;

class Foo implements iFoo
{
    private $id;

    public function getId() {
        return $this->id;
    }

    public function setId($id) {
        $this->id = $id;
    }

    private $name;
```

```
    public function getName() {
        return $this->name;
    }

    public function setName($name) {
        $this->name = $name;
    }

}
```

3.3 Bidjic set to set

Description

Generates multiple output files from multiple input models, using fileset and templateset.

```
+---+       +---+
|   | +---> |   |
|   |       |   |
+---+       +---+
+---+       +---+
|   | +---> |   |
|   |       |   |
+---+       +---+
```

Bidjic task parameters

```
|Attribute  | Description           | Value                | Required      |
|-----------|-----------------------|----------------------|---------------|
|todir      | output dir            | <dirname>            | yes           |
|overwrite  | overwrite output file | true by default      | no            |
|-----------|-----------------------|----------------------|---------------|
|Element    | Description           | Value                | Required      |
|-----------|-----------------------|----------------------|---------------|
|templateset| template set          |                      | yes           |
|fileset    | file set              |                      | if no filelist|
|filelist   | file list             |                      | if no fileset |
```

Example

Generates cpp - c++ classes from Model driven architecture [https://en.wikipedia.org/wiki/Model-driven_architecture] abstraction: bidjim entities.

download [http://sourceforge.net/projects/bidji/files/samples/set2set.tgz/download] set2set.tgz sample.

ant file : build.xml

```
<target name="cpp" description="generate cpp files">
    <bidjic todir="gen" overwrite="true">
        <fileset dir="MyPackage">
            <include name="**/*.bidjim"/>
        </fileset>
        <templateset dir="templates">
            <include name="*.ftl"/>
        </templateset>
    </bidjic>
</target>
```

Note

Output file names are processed from template names:

- *Model*.cpp.ftl + MyPackage/MyModule/Foo.bidjim => *Foo*.cpp
- *Module*.cpp.ftl + MyPackage/MyModule/Foo.bidjim => *MyModule*.cpp

template: Model.h.ftl

```
[#assign entity = bidji.entity]
#ifndef _${PACKAGE}_${MODULE}_${MODEL}_H_
#define _${PACKAGE}_${MODULE}_${MODEL}_H_

namespace ${Package} {
namespace ${Module} {

class ${Model}
{

private:
[#foreach property in entity.property]
    ${property.@type} _${property.@name};
[/#foreach]
```

```
public:
    ${Model}();
    virtual ~${Model}();

public:
[#foreach property in entity.property]
    void set${property.@name?cap_first}(const ${property.@type} & ${property.@
    const ${property.@type} & get${property.@name?cap_first}() const;
[/#foreach]

}; // class ${Model}

[#foreach property in entity.property]
inline const ${property.@type} &
${Model}::get${property.@name?cap_first}() const
{
    return _${property.@name};
}

[/#foreach]

} } // namespace ${Package}::${Module}

#endif // #ifndef _${PACKAGE}_${MODULE}_${MODEL}_H_
```

template: Model.cpp.ftl

```
[#assign entity = bidji.entity]

#include "${Model}.h"

using namespace ${Package}::${Module};

${Model}::${Model}()
[#foreach property in entity.property]
    [#if property.@type == 'int']
, _${property.@name}(-1)
    [#elseif property.@type == 'string']
, _${property.@name}("")
    [#else]
, _${property.@name}()
    [/#if]
[/#foreach]
```

```
{
}

${Model}::~${Model}()
{
}

[#foreach property in entity.property]
void
${Model}::set${property.@name?cap_first}(const ${property.@type} & ${property.
{
    this->_${property.@name} = ${property.@name};
}
[/#foreach]
```

models: MyPackage/MyModule/(Foo|Baz).bidjim

```
<?xml version="1.0"?>
<bidji>
<entity name="Foo">
<property name="id" type="int" />
<property name="name" type="string" />
</entity>
</bidji>
```

result: Foo.h

```
#ifndef _MYPACKAGE_MYMODULE_FOO_H_
#define _MYPACKAGE_MYMODULE_FOO_H_

namespace MyPackage {
namespace MyModule {

class Foo
{

private:
    int _id;
    string _name;

public:
    Foo();
```

```
    virtual ~Foo();

public:
    void setId(const int & id);
    const int & getId() const;
    void setName(const string & name);
    const string & getName() const;

}; // class Foo

inline const int &
Foo::getId() const
{
    return _id;
}

inline const string &
Foo::getName() const
{
    return _name;
}

} } // namespace MyPackage::MyModule

#endif // #ifndef _MYPACKAGE_MYMODULE_FOO_H_
```

result: Foo.cpp

```

#include "Foo.h"

using namespace MyPackage::MyModule;

Foo::Foo()
, _id(-1)
, _name("")
{
}

Foo::~Foo()
{
}

void
Foo::setId(const int & id)
```

```
{
    this->_id = id;
}
void
Foo::setName(const string & name)
{
    this->_name = name;
}
```