VORtech Computing

the Scientific Software Engineers

Postbus 260

2600 AG DELFT

tel. 015-285 0125

fax. 015-285 0126

vortech@vortech.nl

MEMO AS/M12.008

Date 22-02-2012

Author(s) ir. A.M.Sewgobind, ir. R. Kroon, W. de Smit

Subject PBL DeliveryTool Developers QuickStart

**Documentinformatie**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Versie | Auteur | Datum | Opmerkingen | Review |
| 0.1 | RK | 21-02-2012 | Concept | AS |
| 1.0 | AS | 22-02-2012 | First version | JvV |
| 1.1 | WdS | 12-07-2012 | Updated with project setup | AS |
| 1.2 | Rineke Oostenrijk | 07-03-2016 | Mdio description added  Upgraded to new versions: 64bit and current jre |  |
| 1.3 | Rineke Oostenrijk | 02-08-2016 | Param outputformat –of excel|csv Param numberformat –nf decimal|e |  |
| Bestandslocatie: @@nognietingevuld@@ | | | | |

# Introduction

PBLDeliveryTool provides functionality to create Excel reports containing IMAGE data from various model runs, based on an Excel mapping and an Excel template file. It is designed to run from the command line, and a log file is created as well on each run. This document provides a brief introduction aimed at developers.

The tool uses mdioToJava.dll and mdio.jar. Together they form a wrapper to communicate with the MyM libraries (mdios6.dlll, sfs6.dll). Because the tools are so dependent a description of the mdio software is added to this document

# Importing the mdio wrapper and PBLDeliveryTool in Eclipse

If you already have a specific workspace in Eclipse, it is recommended to place the mdio folder and the PBLDeliveryTool folder in your workspace in order to keep the working environment easy to manage.

Import the existing project into your current workspace:

1. Go to *File* **->** *import*.
2. Then *General* -> *Existing projects into workspace*.
3. Now select the directory called java. Eclipse will show projects in the project box. Make sure the checkbox for PBLDeliveryTool and mdioToJAva project are marked and press finish.

***Note****:* The mdio.jar is the jar made of the sources of mdioTojava wrapper. Both projects use the same java e-f /nvironment. The MyM dll’s and the c-wrapper mdioToJava.dll are duplicated in the projects to ensure independency of the projects.

# Configuration of PBLDeliveryTool in Eclipse

## Set Java Build Path

After importing you will see the project in your Package Explorer. Follow the steps below to adjust the Java Build Path:

1. First rightclick on your PBLDeliveryTool project and properties.
2. Then go to *Java Build Path* in the submenu and select the *libraries* tab.
3. A couple of jar’s should be already in the libaries but marked by a red cross. Delete these, and then click on *Add External Jars*.
4. Browse to the subfolder …*/****PBLDeliveryTool/lib*** and select all jars in that folder and add them to the library. The newly added jars should now be added properly and your PBLDeliveryTool project should not give any errors.
5. As a final step go again to the properties of *PBLDeliveryTool*->*libraries* -> open mdio.jar and click on *Native library* -> *location* -> *edit*. Now add from your workspace the directory **/PBLDeliveryTool/lib** to that location.

## Setting up Run/Debug configurations

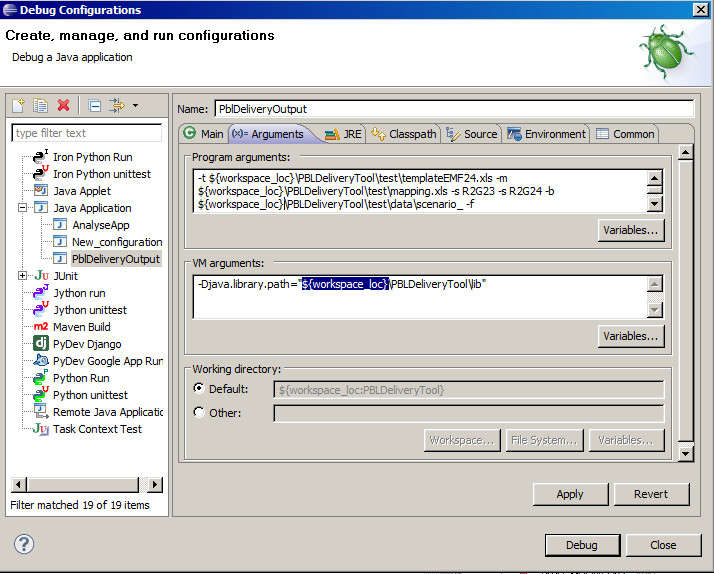
The program can now be build and run without errors. In order to run or debug from within Eclipse you will need to setup run configurations including the proper command line arguments:

1. Rightclick on the *Project* -> *Run as* -> *Run configurations*.
2. Make sure to run it as a Java Application and insert in the main class textfield: *program.PbDeliveryOutput .*
3. Go to the tab *arguments* and insert the program arguments there. Make sure you use absolute paths (apath) in your arguments and capital letters. Path may be partly sustituted by ${workspace\_loc} variable.

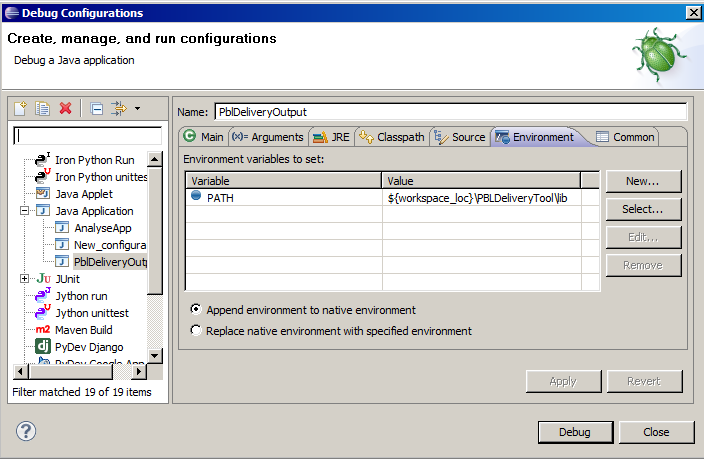
***Example***:

*-f apath\testoutput.xls -l INFO -t apath\template\_test.xls -m root apath\mapping.xls -b apath \scenario\_R2G23 -s R2G23 -r World -r CAN*

1. Add to the field VM Arguments:   
   *-Djava.library.path="${workspace\_loc}\PBLDeliveryTool\lib"*

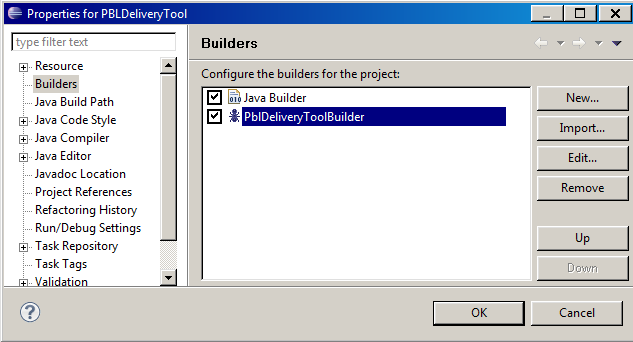


1. *Set PATH in the environment tab, see below*



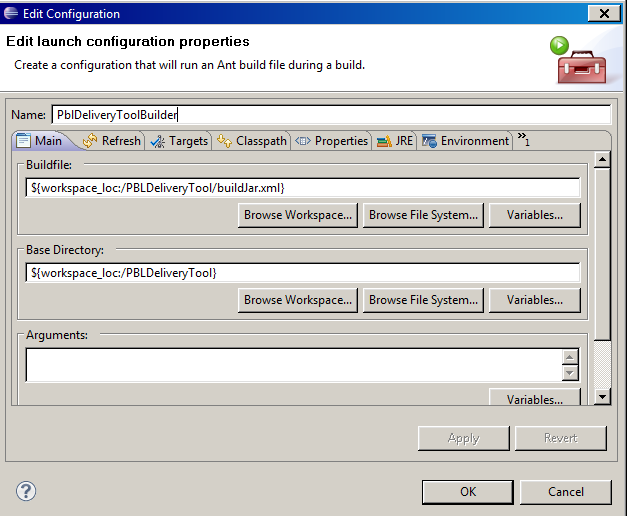
## Creating a jar file out of the project

Create a builder for the project in Eclipse. Right mouse click on the properties of PBLDeliveryTool.



Press new or edit if allready there and fill in the form: Name; Buildfile; Base directory.

Apply and OK. Clean project and build project. The new jar will be available in the <projectdirectory>\jar directory. It is important to use buildjar.xml because it stores the versioninfo otf the build the application.



As alternative you can build the jar from the prompt:

Open command prompt in the <projectdir> :

<ant installdir>\ant -f .\buildJar.xml

Where ant installdir = Y:\netapps\Eclipse\plugins\org.apache.ant\_1.8.2.v20120109-1030\bin\

# System

The program consists of a Java package and supporting native libraries. Because of these native libraries it is important that the Java runtime environment has the same platform as these libraries. The delivery consists of:

* PblDeliveryTool.jar Encapsulates all java code
* mdioToJava.dll Provides the JNI bridge
* mdio6.dll DLL for actual I/O
* sf6.dll Required support dll

Note: The dll’s are dependent of visual studio vc…120.dll s ( in case of VC12). So with any c-compilerversion change the dlls must be recompiled.

# Installation

Due to the native libraries, installation is slightly less straight forward than with a normal Java archive as delivery, i.e. the PATH variable must point to the extra libraries.

Example commands when the software is installed in C :\myPbl

C:\myPbl>Set PATH=%PATH%;c:\myPbl

C:\myPbl>"C:\Program Files\Java\jre7\bin\java" -jar PblDeliveryTool.jar -t

C:\vortech\PBL\test\templateEMF24.xls -m C:\vortech\PBL\test\mapping.xls -s R2G

23 -s R2G24 -b C:\vortech\PBL\test\data\scenario\_ -f c:\vortech\PBL\test.xls

NOTE: Replace the arguments with files/paths on your own system and adjust the java path to the current version . Including Java in the system PATH reduces the call to: java –jar ….

# Functional flow

A functional overview of the program and its data flow is shown in Figure 1.



Figure 1 Functional overview and data flow

The program uses this order to create a report:

1. Parse command line options ( main in PblDeliveryTool )
2. Read Excel mapping file ( creates a Mapping object)
3. Create output document / read template file ( see OutputDocument.setupDocumentFromTemplate)
4. Find a mapping for each template and (class Engine / doWork()):
   1. Open the corresponding M-file
   2. Read timesteps that are present in the template
      1. Filter these to only keep the relevant regions (class MFileFilter, RegionControl)
      2. Write the filtered rows to the OutputDocument

# Design

The program has two packages that make up the program, a section data for objects that holds data and a program structure that contains the actual algorithms to create a report.

The algorithmic part is relatively simple, as most work is done by the class ‘Engine’, with a helper class to filter the data coming from the M-file.

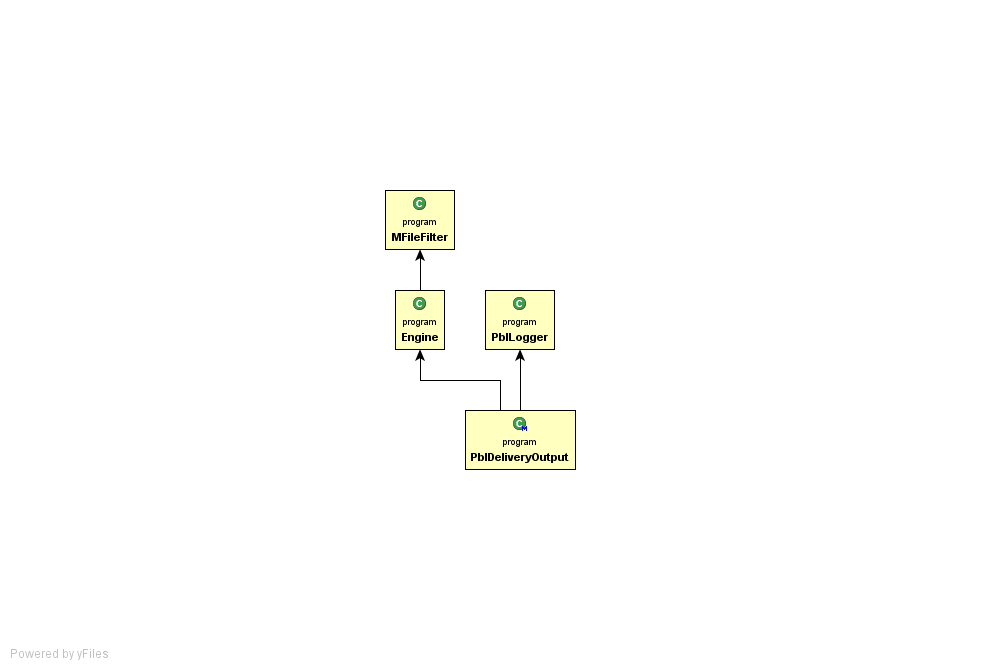


Figure 2 Algorithmic part of program

The data part is slightly more complicated due to the types that control the output. The mapping options that are present as class:

* ‘global, noglobal, regional’ are defined inWorldType.
* The number of dimensions defines the IMAGE/TIMER type and is contained in RegionType.ModelType
* The region names and an entry ‘World’ for global are stored in RegionType.RegionName

The template entries are read and stored in a class Mapping, consisting mainly of row objects holding the information for each row in the template. Both output and template rows are represented by the OutputRow class which has a relation to the OutputDocument class which in turn writes the output file. Additional helper functions, for example to skip the empty region when having 28 dimensions in a file are in RegionControl.

The resulting diagram (showing direct & indirect dependencies):

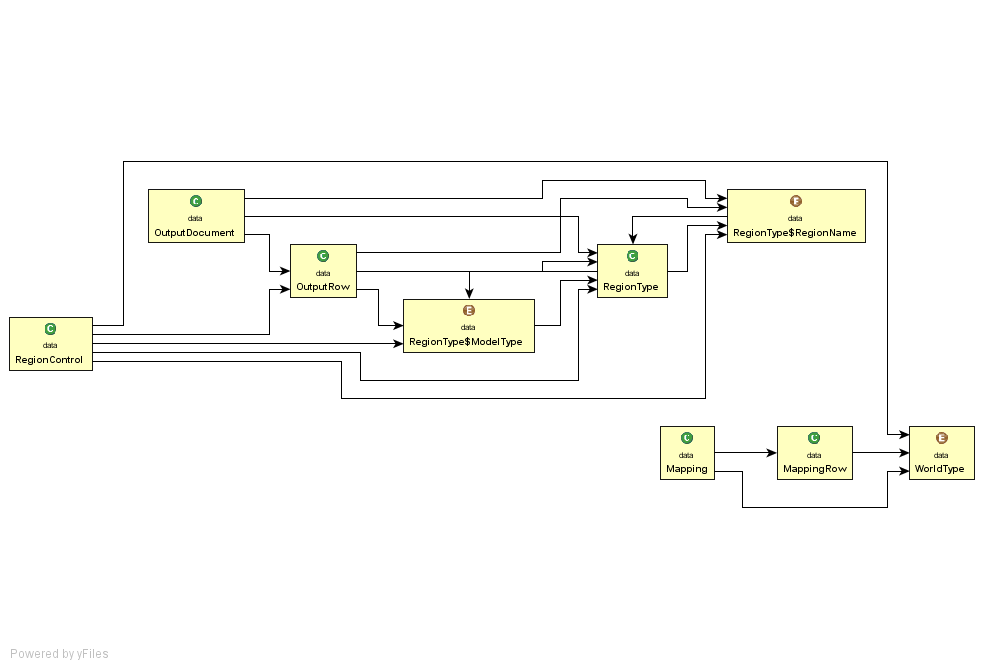


Figure 3 Data part of program. The Engine class holds an OutputDocument and a Mapping object and uses these to create the report.