

Release Notes

**XMLMill for Java - XMLMill for Domino**

**2.70**



xmlmill x

xmlmill x

© 2000-2006 Pecunia Data Systems bvba. All rights reserved.

NOTICE: All information contained herein is the property of Pecunia Data Systems bvba.

This publication and the information herein are furnished AS IS, are subject to change without notice, and should not be construed as a commitment by Pecunia Data Systems bvba. Pecunia Data Systems bvba assumes no responsibility or liability for any errors or inaccuracies, makes no warranty of any kind (express, implied, or statutory) with respect to this publication, and expressly disclaims any and all warranties of merchantability, fitness for particular purposes, and noninfringement of third-party rights.

# Table of Contents

<b>1. Preface.....</b>	<b>2</b>
<b>2. Upgrading from previous versions.....</b>	<b>3</b>
2.1. Upgrading from version 2.60.....	3
2.2. Upgrading from earlier versions.....	3
<b>3. What's new ?.....</b>	<b>4</b>
3.1. XMLMill for Java.....	4
3.2. XMLMill for Domino.....	4
<b>4. XMLMill for Java.....</b>	<b>5</b>
4.1. Multi-column feature.....	5
4.1.1. Multiple columns on one page.....	5
4.1.2. Multiple column segments on one page.....	5
4.1.3. Spanning elements.....	5
4.2. XMLMill and Ant.....	6
4.2.1. Configuration.....	6
4.2.1.1. Create a classpath reference.....	6
4.2.1.2. Create an XMLMill task.....	6
4.2.2. Generating PDF documents.....	6
4.2.3. Comparing PDF documents.....	7
4.2.4. Parameters.....	7
4.2.4.1. <mill> element.....	7
4.2.4.2. <generate> element.....	8
4.2.4.3. <compare> element.....	9
4.2.5. Parameters as nested elements.....	9
4.2.5.1. Inside a <mill> element.....	9
4.2.5.1.1. <generate> element.....	9
4.2.5.1.2. <compare> element.....	10
4.2.5.1.3. <classpath>.....	10
4.2.5.1.4. <jvmarg>.....	10
4.2.5.2. Inside a <generate> element.....	10
4.2.5.2.1. <fileset> element.....	10
4.2.5.3. Inside a <compare> element.....	10
4.2.5.3.1. <master> element.....	10
4.2.5.3.2. <slave> element.....	10
4.2.5.4. Inside a <master> element.....	10
4.2.5.4.1. <fileset> element.....	10
4.2.5.5. Inside a <slave> element.....	10
4.2.6. Logging.....	10
4.2.6.1. Generation process.....	10
4.2.6.1.1. Summary log.....	10
4.2.6.1.2. Detailed log.....	11
4.2.6.2. Compare process.....	11
4.2.6.2.1. Summary log.....	11
4.2.6.2.2. Detailed log.....	12
4.2.7. Examples.....	12
4.3. Background-images behavior.....	13
<b>5. XMLMill for Domino.....</b>	<b>14</b>

6. Bugfixes..... 15



## 1. Preface

The XMLMill application is intended for software developers who want to generate .pdf documents from xml and/or xsl.

For an overview of how to use XMLMill in general please consult in the **docs/** directory:.

- ◆ **apidoc/** -- The JavaDoc concerning the **XMLMill** public api.
- ◆ **userguide.pdf** -- XMLMill user's guide.
- ◆ **dtdguide.pdf** -- An explanation of XMLMill's elements (tags) and their attributes.
- ◆ **samplesguide.pdf** -- An overview of the examples in the **samples/** directory contained in the download.
- ◆ **digitalsignatures.pdf** -- An overview of how to digitally sign PDF documents (one pass generation and signing).

If you have questions, please do not hesitate to send a mail to [support@xmlmill.com](mailto:support@xmlmill.com).

☞ *This document is completely generated with XMLMill 2.70 using **rnotes270.xml** and **rnotes.xsl**. These files can be found in the directory **samples/docs/rnotes** in the download.*



## 2. Upgrading from previous versions

### **2.1. Upgrading from version 2.60.**

The behavior of the background-image positioning has changed, which need a small change in your code and/or images used. Check the [background-image bahvior](#) section for more information.

Use the `xmlmill.xsd` of this version in order to be able to use the new functionalities.

### **2.2. Upgrading from earlier versions**

The best procedure to make your transition as smooth as possible is as follows:

1. Back up all your xml/xsl/mill files or Java code.
2. Read all release notes <http://www.xmlmill.com/news.html> from your current version to the latest version in correct order (from oldest release notes to newest).
3. Modify your xml/xsl/mill files or Java code accordingly to the release notes.

☞ *Previous versions can be found at <http://www.xmlmill.com/download/xmlmillforjava.html#pv> or <http://www.xmlmill.com/download/xmlmillfordomino.html#pv>*



## 3. What's new ?

### 3.1. XMLMill for Java

This release notes document describes all new (or enhanced) functionalities of this release. The main changes are:

- ◆ Multi-column feature.
- ◆ Support for using XMLMill with Ant.
- ◆ Implement the **orphans** attribute for the `<ml:paragraph>` element.
- ◆ Implement the **background-repeat** attribute for the `<ml:region-*>` elements.
- ◆ Bug fixes.

### 3.2. XMLMill for Domino

This release notes document describes all new (or enhanced) functionalities of this release. The main changes are:

- ◆ Multi-column feature.
- ◆ Implement the **orphans** attribute for the `<paragraph>` element.
- ◆ Implement the **background-repeat** attribute for the `<region-*>` elements.
- ◆ Bug fixes.

xmlmill x

xmlmill x

xmlmill x

## 4. XMLMill for Java

### 4.1. Multi-column feature

As of this version XMLMill supports multiple columns on a page. Moreover, a page can be divided in different segments containing each a different number of columns.

#### 4.1.1. Multiple columns on one page

In order to define multiple columns on a page the **column-count** attribute needs to be defined at the **<region-body>** element.

The width between the columns is defined using the **column-gap** attribute.

```
<ml:region-body column-count="2" column-gap="12pt"/>
```

The above example defines 2 columns on a region-body with a gap between the columns of 12 points.

- ☞ *Multiple columns can only be defined at the **<region-body>** element.*
- ☞ *For more information regarding the **column-count** and **column-gap** attributes please consult the **dtdguide.pdf** document.*
- ☞ *For examples consult the **samples/mill** directory (**Attribute\_column-count-\*.mill** files).*

#### 4.1.2. Multiple column segments on one page.

Sometimes it is necessary to have a different number of columns on a page. This can be achieved using the **<flow-segment>** element within a **<flow>** element.

```
<ml:flow flow-name="xsl-region-body">
  <ml:flow-segment overflow="error-if-overflow" height="50%" segment-name="seg1"
    column-count="2">
    ...
  </ml:flow-segment>
</ml:flow>
```

Above a flow-segment is defined having 2 columns.

- ☞ *For more information regarding the **flow-segment** element please consult the **dtdguide.pdf** document.*
- ☞ *For examples consult the **samples/mill** directory (**Attribute\_column-count-\*.mill** files).*

#### 4.1.3. Spanning elements

It is possible to span an element over multiple columns by using the **span** attribute. This attribute applies to following elements:

- ◆ **ml:barcode**
- ◆ **ml:external-graphic**
- ◆ **ml:leader**
- ◆ **ml:signature**
- ◆ **ml:table**
- ◆ **ml:table-and-caption**
- ◆ **ml:textbox**

☞ For more information regarding the **span** attribute please consult the **dtdguide.pdf** document.

☞ For examples consult the **samples/mill** directory.

## 4.2. XMLMill and Ant.

As of this version XMLMill can be used with Ant in order to :

- ◆ Generate PDF documents
- ◆ Compare PDF documents

☞ A basic knowlegde of creating build files in Ant is assumed.

### 4.2.1. Configuration

#### 4.2.1.1. Create a classpath reference

Create a classpath reference for the XMLMill task. It must include all **.jar** files that XMLMill uses, including the **xmlmill.jar** itself. In case JSDK1.3 is used, following applies:

```
<path id="mill-classpath">
  <fileset dir="C:\Data\XMLMill\Release-to-production\_Libraries">
    <include name="dom.jar" />
    <include name="jaxp-api.jar" />
    <include name="sax.jar" />
    <include name="xalan.jar" />
    <include name="xercesImpl.jar" />
    <include name="xmlmill.jar" />
    <include name="log4j-1.2.8.jar" />
  </fileset>
</path>
<property name="mill.path" refid="mill-classpath" />
```

☞ Above example assumes JSDK 1.3 is used.

In case JSDK 1.4 or JSDK 1.5 is used following applies:

```
<path id="mill-classpath">
  <fileset dir="C:\Data\XMLMill\Release-to-production\_Libraries">
    <include name="xmlmill.jar" />
    <include name="log4j-1.2.8.jar" />
  </fileset>
</path>
<property name="mill.path" refid="mill-classpath" />
```

#### 4.2.1.2. Create an XMLMill task

Create an XMLMill task in your build file using the **<taskdef>** task:

```
<taskdef name="mill"
  classname="com.xmlmill.connector.ant.XMLMillTask" ❶
  classpath="{mill.path}" /> ❷
```

❶ The **public** class to use is: **com.xmlmill.connector.ant.XMLMillTask**.

❷ Refer to the **mill-classpath**.

### 4.2.2. Generating PDF documents

To generate PDF documents the **<generate>** element in the **<mill>** task is used:

```
<mill maxmemory="512m"
  tempdir="c:\bro\anttest" ❶
  printsummary="on"
```

```

    loglevel="debug">
<classpath refid="mill-classpath"/>
<!--
  Generate the PDF documents
-->
<generate name="G1" ❷
    haltonwarning="false"
    outputfolder="C:\brol\outputtest"
    validate="on"
    todir="C:\data\generate-1.log">
  <fileset dir="C:\Data\_unittests"> ❸
    <include name="mill/*.mill"/>
  </fileset>
</generate>
</mill>

```

- ❶ The attributes defined at **<mill>** level are automatically inherited by the descendant elements (**<generate>** and **<compare>**) if applicable.
  - ❷ For more information regarding the applicable attributes please consult the [Parameters](#) section.
  - ❸ Multiple **<fileset>** elements can be defined to determine which **.mill** or **.xml** files to handle.
- ☞ A **<mill>** task can contain multiple **<generate>** elements.
- ☞ The generation of PDF documents defined by each **<generate>** element is executed in a separate **JVM** environment.

### 4.2.3. Comparing PDF documents

To compare PDF documents the **<compare>** element in the **<mill>** task is used:

```

<mill maxmemory="512m"
    tempdir="c:\brol\anttest" ❶
    printsummary="on"
    loglevel="debug">

  <classpath refid="mill-classpath"/>
  <!--
    Compare the PDF documents
  -->
  <compare name="C1" ❷
    haltonerror="false"
    todir="C:\XMLMill2.60\samples\mill\log\compare-1.log" >
    <master> ❸
      <fileset dir="C:\Data\XMLMill\TestDirectory\_master">
        <include name="mill/**/*.*.pdf"/>
      </fileset>
    </master> ❹
    <slave dir="C:\brol\outputtest"/>
  </compare>
</mill>

```

- ❶ The attribute defined at **<mill>** level are automatically inherited by the descendant elements (**<generate>** and **<compare>**) if applicable.
  - ❷ For more information regarding the applicable attributes please consult the [Parameters](#) section.
  - ❸ The **<master>** element defines which PDF document should act as 'master'. These are the correct documents to be compared to. It must contain exactly 1 **<fileset>** element.
  - ❹ The **<slave>** element defines where the generated documents are stored.
- ☞ A **<mill>** task can contain multiple **<compare>** elements.
- ☞ The comparing of PDF documents defined by each **<compare>** element is executed in a separate **JVM** environment.

### 4.2.4. Parameters

#### 4.2.4.1. <mill> element

- ☞ The parameters defined at **<mill>** level are automatically inherited by the descendant

elements (`<generate>` and `<compare>`) if applicable.

Attribute	Description	Required
<code>append</code>	Append messages to an existing log file instead of overwriting the log file. If the log file does not exist, it is created.	No; default is <b>off</b> .
<code>config</code>	Defines the configuration file ( <code>config.xml</code> ) to use. If not defined, the The default configuration file in the <code>xmlmill.jar</code> file is used.	No.
<code>dir</code>	The directory in which to invoke the VM (if <code>jvm</code> attribute is used).	No
<code>haltonerror</code>	Stop the build process if an error occurs during the generation or comparing of pdf documents.	No; default is <b>off</b> .
<code>haltonwarning</code>	Stop the build process if a warning occurs during the generation of pdf documents.	No; default is <b>off</b> .
<code>jaxp</code>	Defines the JAXP version of the underlying implementation ( <code>1.1</code> , <code>1.2</code> or <code>1.3</code> ).	No; default <b>1.2</b> .
<code>jvm</code>	The command used to invoke the Java Virtual Machine, default is 'java'. The command is resolved by <code>java.lang.Runtime.exec()</code> .	No; default is <b>java</b> .
<code>loglevel</code>	Defines the loglevel. The values can be: DEBUG, INFO, WARN, ERROR, FATAL, ALL, OFF. If the value passed is different than the ones defined here, the loglevel is set to DEBUG.	No; default is <b>INFO</b> .
<code>includeantruntime</code>	Implicitly add the Ant classes required to run the generation and competing process to the classpath.	No; default is <b>true</b> .
<code>maxmemory</code>	Maximum amount of memory to allocate to the forked VM.	No
<code>outputfolder</code>	Defines the outputfolder. This is where the generated PDF documents are stored.	YES
<code>printsummary</code>	Print one-line statistics.	No; default is <b>off</b> .
<code>size</code>	Define the minimum size (in bytes) the pdf document should be.	No; default is the project's base directory.
<code>tempdir</code>	Where temporary files should be put. During the generate and compare processes temporary files are generated concerning the selected files and attribute settings.	No; default is the project's base directory.
<code>timeout</code>	Cancel the individual generation of PDF documents if they don't finish in the given time (measured in milliseconds).	No
<code>todir</code>	Directory to write the reports to.	YES
<code>validate</code>	Validates the result tree. <b>Note: This requires a JAXP 1.3 compliant implementation of the underlying transformer.</b>	No; default is <b>off</b> .
<code>verbose</code>	Send any output generated to the console as well as to the logfile(s).	No; default is <b>off</b> .
<code>xslt</code>	XSLT transformation only (transform .xml/.xsl file to a .mill file).	No; default is <b>off</b> .

#### 4.2.4.2. `<generate>` element

Attribute	Description	Required
<code>append</code>	Append messages to an existing log file instead of overwriting the log file. If the log file does not exist, it is created.	No; default is <b>off</b> .
<code>config</code>	Defines the configuration file ( <code>config.xml</code> ) to use. If not defined, the The default configuration file in the	No.

Attribute	Description	Required
	<b>xmlmill.jar</b> file is used.	
<b>haltonerror</b>	Stop the build process if an error occurs during the generation or comparing of pdf documents.	No; default is <b>off</b> .
<b>jaxp</b>	Defines the JAXP version of the underlying implementation ( <b>1.1</b> , <b>1.2</b> or <b>1.3</b> ).	No; default <b>1.2</b> .
<b>loglevel</b>	Defines the loglevel. The values can be: DEBUG, INFO, WARN, ERROR, FATAL, ALL, OFF. If the value passed is different than the ones defined here, the loglevel is set to DEBUG.	No; default is <b>INFO</b> .
<b>name</b>	Defines the name of the <compare> element. This name is used as a reference in the log file (see the <a href="#">Logging</a> section).	
<b>printsummary</b>	Print one-line statistics.	No; default is <b>off</b> .
<b>size</b>	Define the minimum size (in bytes) the pdf document should be.	No; default is the project's base directory.
<b>todir</b>	Directory to write the reports to.	YES
<b>validate</b>	Validates the result tree. <b>Note: This requires a JAXP 1.3 compliant implementation of the underlying transformer.</b>	No; default is <b>off</b> .
<b>verbose</b>	Send any output generated to the console as well as to the logilfe(s).	No; default is <b>off</b> .
<b>xsl</b>	Defines the .xsl file that will be used to transform the .xml files. If specified this .xsl file will overrule any .xsl file defined in the .xml file (defined in the <?xml-stylesheet ...?>ag).	No.
<b>xslt</b>	XSLT transformation only (transform .xml/.xsl file to a .mill file).	No; default is <b>off</b> .

#### 4.2.4.3. <compare> element

Attribute	Description	Required
<b>append</b>	Append messages to an existing log file instead of overwriting the log file. If the log file does not exist, it is created.	No; default is <b>off</b> .
<b>haltonerror</b>	Stop the build process if an error occurs during the generation or comparing of pdf documents.	No; default is <b>off</b> .
<b>loglevel</b>	Defines the loglevel. The values can be: DEBUG, INFO, WARN, ERROR, FATAL, ALL, OFF. If the value passed is different than the ones defined here, the loglevel is set to DEBUG.	No; default is <b>INFO</b> .
<b>name</b>	Defines the name of the <compare> element. This name is used as a reference in the log file (see the <a href="#">Logging</a> section).	No
<b>printsummary</b>	Print one-line statistics.	No; default is <b>off</b> .
<b>todir</b>	Directory to write the reports to.	YES
<b>verbose</b>	Send any output generated to the console as well as to the logilfe(s).	No; default is <b>off</b> .

#### 4.2.5. Parameters as nested elements

##### 4.2.5.1. Inside a <mill> element

##### 4.2.5.1.1. <generate> element

---

Multiple **<generate>** elements can be defined inside the **<mill>** element.

#### 4.2.5.1.2. <compare> element

Multiple **<compare>** elements can be defined inside the **<mill>** element.

#### 4.2.5.1.3. <classpath>

The **<mill>** task supports a nested **<classpath>** element that represents a PATH like structure.

#### 4.2.5.1.4. <jvmarg>

Additional parameters may be passed to the new VM via nested **<jvmarg>** elements. For example:

```
<mill">
  <jvmarg value="-Djava.compiler=NONE" />
  .
.</mill>
```

would run the test in a VM without JIT.

**<jvmarg>** allows all attributes described in [Command-line Arguments](#).

### 4.2.5.2. Inside a <generate> element

#### 4.2.5.2.1. <fileset> element

Multiple **<fileset>** elements can be defined inside the **<generate>** element.

### 4.2.5.3. Inside a <compare> element

#### 4.2.5.3.1. <master> element

Exactly one **<master>** element can be defined inside the **<compare>** element.

#### 4.2.5.3.2. <slave> element

Exactly one **<slave>** element can be defined inside the **<slave>** element.

### 4.2.5.4. Inside a <master> element

#### 4.2.5.4.1. <fileset> element

Multiple **<fileset>** elements can be defined inside the **<generate>** element.

### 4.2.5.5. Inside a <slave> element

No sub elements.

## 4.2.6. Logging

The logging reports produced are different for the generation or comparing process:

### 4.2.6.1. Generation process

For each generation process 2 log files are generated:

#### 4.2.6.1.1. Summary log

A summary file with an overview of the PDF documents to generate, the number of PDFs successfully generated and the number of failures, followed by a list of all failed **.xml/.mill** files.

The name of this summary file is the same as defined by the **todir** attribute of the **<generate>**

---

element.

```
Job           :Example_volume_paragraph ❶  
Date          :Thu Jul 27 22:19:14 CEST 2006 ❷  
Details       :E:\Data\tests\reports\details\volume-paragraph-1154031508062.log ❸  
PDFs          :3 ❹  
Success       :3 ❺  
Failures      :0 ❻  
Time elapsed:45,719 ❼
```

This summary files contains:

- ❶ The name of the 'Job' (this is the name of the process as defined by the **name** attribute of the **<generate>** element.
- ❷ The date when the file was created.
- ❸ The location of the log file containing all details of the generation process.
- ❹ The number of PDF documents to generate.
- ❺ The number of successfull generated PDF documents.
- ❻ The number of failed documents.
- ❼ The elapsed time.

#### 4.2.6.1.2. Detailed log

A detailed log file containing all the logging messages of the generation process (as produced by XMLMill)

```
INFO [main]: =====  
INFO [main]: = XMLMill build: XMLMill Java @version@  
INFO [main]: = Licensed to : XMLMill (Aartselaar - Belgium) (0000000)  
INFO [main]: = ErrorLog created at: Thu Jul 27 22:18:28 CEST 2006  
INFO [main]: =====  
INFO [main]: Current log file:  
E:\Data\tests\reports\details\volume-paragraph-1154031508062.log  
INFO [main]: Current log-level: INFO  
...
```

The detailed log file contains all logging messages (based on the **log-level** defined) of the generation process as defined by XMLMill.

#### 4.2.6.2. Compare process

For each compare process 2 log files generated:

##### 4.2.6.2.1. Summary log

A summary file with an overview of the PDF documents to compare, the number of PDFs successfully compared and the number of failures, followed by a list of all failed **.pdf** files.

The name of this summary file is the same as defined by the **todir** attribute of the **<compare>** element.

```
Job           :C1 ❶  
Date          :Thu Jul 27 22:27:21 CEST 2006 ❷  
Details       :E:\Data\tests\reports\details\compare-1-1154031965343.log ❸  
PDFs          :178 ❹  
Success       :178 ❺  
Failures      :0 ❻  
Time elapsed:76,095 ❼
```

This summary files contains:

- ❶ The name of the 'Job' (this is the name of the process as defined by the **name** attribute of the

<generate> element.

- ② The date when the file was created.
- ③ The location of the log file containing all details of the compare process.
- ④ The number of PDF documents to compare (as defined by the <master> element).
- ⑤ The number of successful compare PDF documents.
- ⑥ The number of failed compares.
- ⑦ The elapsed time.

#### 4.2.6.2.2. Detailed log

A detailed file containing all the logging messages of the compare process.

```
INFO [main]: =====
INFO [main]: = XMLMill build: XMLMill Java @version@
INFO [main]: = Licensed to : XMLMill (Aartselaar - Belgium) (0000000)
INFO [main]: = ErrorLog created at: Thu Jul 27 22:26:05 CEST 2006
INFO [main]: =====
INFO [main]: Current log file:
E:\Data\XMLMill\Release-to-production\rev2-70-65\tests\reports\details\compare-1-115403
1965343.log
INFO [main]: Current log-level: DEBUG
INFO [main]: Comparing... E:\Data\...\Attribute_topleft.pdf with
E:\Data\..output\Attribute_topleft.pdf
INFO [main]: Comparing... E:\Data\...\Attribute_text-transform.pdf with
E:\Data\...\Attribute_text-transform.pdf
...
```

#### 4.2.7. Examples

Following example is taken out of an existing build file. During the build process PDF documents are automatically generated and compared with a 'master' directory.

```
<mill config="${testdirectory.dir}/config/config.xml" ❶
  printsummary="on"
  loglevel="info"
  tempdir="${test.temp}"
  verbose="off"
  haltonwarning="false"
  maxmemory="512m"
  haltonerror="no"
>
  <classpath>
    <path refid="mill-classpath"/> ❷
  </classpath>
  <!--
  Generate the PDF documents
  -->
  <generate name="Generic" ❸
    outputfolder="${test.results}"
    validate="on"
    todir="${test.reports}/generic.txt"
  >
    <fileset dir="${testdirectory.dir}/_unittests"> ❹
      <include name="**/*.mill"/>
      <include name="**/*.xml"/>
      <exclude name="**/volume/**"/>
      <exclude name="**/nissan/**"/>
      <exclude name="**/mill/Element_Signature-*.mill"/>
    </fileset>
  </generate>

  <generate name="Nissan" ❺
    outputfolder="${test.results}"
    validate="on"
    todir="${test.reports}/nissan.txt"
    append="true"
    xsl="${testdirectory.dir}/_unittests/customers/nissan/belspec-2.00.xsl"
  >
    <fileset dir="${testdirectory.dir}/_unittests">
      <include name="**/nissan/*.xml"/>
    </fileset>
```

```

</generate>

<!--
Compare the PDF documents
-->
<compare loglevel="debug"
        name="C1"
        verbose="off"
        todir="${test.reports}/compare-1.txt"
>
  <master>
    <fileset dir="${testdirectory.dir}/_master">
      <include name="**/*.pdf"/>
    </fileset>
  </master>
  <slave dir="${test.results}"/>
</compare>
</mill>

```

- ❶ The **<mill>** element defines the configuration file to use (used for generating the pdf document) and set the memory to use to 512 MB.
  - ❷ Defines the classpath (see the [Create a classpath reference](#) section for more information).
  - ❸ Define the first batch of PDF document to generate, based on the **<fileset>**s defined.
  - ❹ The **<fileset>** defines the location of the **.mill** or **.xml** files. The **.xml** files must include a reference to their respective **.xsl** files.
  - ❺ The generate element defines **xml** files that needs to be generated with the same **.xsl** file (as defined by the **xsl** parameter).
  - ❻ The compare element defines the location of the master pdf documents and the location of the slave PDF documents to compare with.
- ☞ *If there are more documents in the slave directory than in the master that will not be reported as an error.*
- ☞ *if an **.xsl** file is define using the **xsl** parameter, it will overrule any **.xsl** file referenced in an **.xml** document.*

### 4.3. Background-images behavior

As of this version the XMLMill supports the **background-repeat** attribute.

A region can have a background image that can be repeated over the region horizontally, vertically, both or not at all.

As the **background-repeat** has a default value of '**repeat**' the images will automatically be repeated over the region (in previous versions the background-image was not repeated - as it was not implemented). Please modify your code if needed.

The background-image is no longer resized to fit into the region. If an images is too big to fit into the region it will automatically be clipped so it will not overflow to another region. Please resize your image if needed.

Above behavior changes are implemented to become more compliant with the FO standard.

- ☞ *Consult the **background-repeat** attribute in the **dtdguide.pdf** document for more information regarding this attribute.*
- ☞ *Please consult the examples in the **samples/mill** directory (**Attribute\_background-image-\*.mill** and **Attribute\_background-color-\*.mill** files).*

## 5. XMLMill for Domino

Please visit the previous sections in the previous chapter.



## 6. Bugfixes

Following bugfixes were solved:

- ◆ In case the attribute **file** has not been defined at **<document>** level, the name of the generated PDF document is the same as the name of the xml document.
- ◆ The base-directory was wrong defined in case the xsl file was in another directory than the xml file (the base directory should always be the directory where the xsl-file is located).
- ◆ The base-directory was wrong defined in case the xsl file was defined in the xml-file (the base directory should always be the directory where the xsl-file is located).
- ◆ The length and location of a **<leader>** element was not always correctly calculated.
- ◆ As of this version the **orphans** attribute is implemented for the **<paragraph>** element.

