Documentation of JavaTemplates

Projekt: JavaTemplates

Version: 2001-12-18, Draft-Version

URL:

Inhaltsverzeichnis

1 SourceForge

Hosted on SourceForge Hosted on SourceForge

1.1 Download

Download from SourceForge www http://sourceforge.net/project/showfiles.php?group_id=40873

2 Introduction

JavaTemplates is a little but very powerful tool for automatical generation of files (source code, HTML-code, ...). It's based on in my opinion finest template mechanism ever exists: JavaServerPages (JSP)!
The basic idea is, that you define your project, data structures, document, article or what ever you want, in a XML file. The XML file will be rendered with JavaTemplates to the format you want (e. g. HTML, java, fortran, SQL, LaTeX). You can compare JavaTemplates / JSP with XSLT. The advantage of JavaTemplates / JSP is the full support of Java functionalities. It's on your own to use XSLT or JavaTemplates depending on the problem you want to solve (see FAQ).
This document contains all important informations for using JavaTemplates as generator for documentations (HTML, PDF, ...) and big software projects.

2.1 Examples

Let's give us some examples:

- Generate your documentation (e. g. this document is rendered by JavaTemplates). Use HTML for online presentation and e. g. PDFLaTeX for printable PDF documents (or StarOffice 6.0, RTF and so on).Generate your documentation (e. g. this document is rendered by JavaTemplates). Use HTML for online presentation and e. g. PDFLaTeX for printable PDF documents (or StarOffice 6.0, RTF and so on).
- Generate Emails with dynamic content within your application. You can customize the Email by editing the jsp template file.Generate Emails with dynamic content within your application. You can customize the Email by editing the jsp template file.
- Generate your HTML pages with the advantage of getting pages conform to your design styles defined at one place.Generate your HTML pages with the advantage of getting pages conform to your design styles defined at one place.
- Develop your great software project by defining your basic structures and generation with JavaTemplates: The result is a homogen structured application. You want to change the database or EJB-Container? No problem: Adapt your central template files and enjoy in your hobbies instead of doing lot of dirty copy and paste.Develop your great software project by defining your basic structures and generation with JavaTemplates: The result is a homogen structured

application. You want to change the database or EJB-Container? No problem: Adapt your central template files and enjoy in your hobbies instead of doing lot of dirty copy and paste.

3 Getting started

3.1 What do I need?

1. You need Java 1.2 or later (JAVA_HOME must in some cases be set).You need Java 1.2 or later (JAVA_HOME must in some cases be set).
2. You need Jakarta-ANT. (ANT_HOME must in some cases be set). You need optional jakarta-ant-*-optional.jar in the lib directory of installed ANT.You need Jakarta-ANT. (ANT_HOME must in some cases be set). You need optional jakarta-ant-*-optional.jar in the lib directory of installed ANT.
3. JavaTemplates must be in your classpath:JavaTemplates must be in your classpath:

```
1  TEMPL\_HOME=<your\_path\_to\_Java\_Templates>
2  CLASSPATH=‘find TEMPL\_HOME/lib/ -name \*.jar | tr "’\n"’ : ‘TEMPL\_HOME/build/cl
3  export CLASSPATH
```

3.2 First test

1. Go first in the directory src/DocGen of JavaTemplates.Go first in the directory src/DocGen of JavaTemplates.
2. Type simplyType simply

   `ant`

   ..
3. Watch the source JavaTemplatesDoc.xml and results in doc/html directory of JavaTemplates.Watch the source JavaTemplatesDoc.xml and results in doc/html directory of JavaTemplates.

3.3 Hello world

The following example you can find in directory examples/HelloWorld of the JavaTemplates directory.

1. Let's write a very simple XML file:Let's write a very simple XML file:

Dateiname: HelloWorld.xml

```
1  <?xml version="’1.0"’ encoding="’ISO-8859-1"’ ?>
2  <message>
3    Hello World.
4  </message>
```

2. Let's write a build.xml file needed by ANT:Let's write a build.xml file needed by ANT:

Dateiname: build.xml

```
<project name="'HelloJavaTemplates"' default="'render"' basedir="'."'>
  <target name="'render"' >
      <taskdef name="'JavaTemplates"' classname="'de.micromata.templ.JavaTemplatesForAnt"
        <JavaTemplates scratchDir ="'."'
                       outDir="'."'
                       force ="'true"'
                       validate="'true"'
                       templateFile="'HelloWorld.jsp"'
                       inFile="'HelloWorld.xml"'
                       outFile="'HelloWorld.java"'/>
  </target>
</project>
```

3. Now let's write the jsp file for rendering our HelloWorld.java:Now let's write the jsp file for rendering our HelloWorld.java:

Dateiname: HelloWorld.jsp

```
<%@ page import="'java.util.*,java.io.*,de.micromata.templ.*"'%><%
  TNode rootNode = (TNode)session.getValue("'TNode"');
  String message = rootNode.getValue().trim();
%>/**
 * An example of generating source code.
 *
 */

public class HelloWorld
{
  public static void main(String args[])
  {
    System.out.println("'<%=message%>"');
  }
}
```

4. Let's go:Let's go:

```
kai@sokoban:~/Micromata/JavaTemplates/examples/HelloWorld> ant
kai@sokoban:~/Micromata/JavaTemplates/examples/HelloWorld> export CLASSPATH=.:CLA
kai@sokoban:~/Micromata/JavaTemplates/examples/HelloWorld> javac HelloWorld.java
kai@sokoban:~/Micromata/JavaTemplates/examples/HelloWorld> java HelloWorld
Hello World.
kai@sokoban:~/Micromata/JavaTemplates/examples/HelloWorld>
```

3.4 Writing dynamic Emails in your Java Application

A example for composing emails and sending them via javamail. (Make copy and paste of ProjectForge)

3.5 Writing big web applications

Wolle's light version of ProjectForge.

4 Technology

JavaTemplates uses DOM for rendering XML files and converts them to a simple tree structure model (TNodes). DOM isn't very userfriendly for using it directly. We have also integrate JDOM instead but not distributed it.
For rendering the jsp files JavaTemplates uses jasper of tomcat. The XML-tree will be put in the user's session. For using thes rendering engine without starting a servlet engine like tomcat, JavaTemplates fakes all needed classes like HttpRequest, HttpSession and so on.
For more comfort in using TNodes in your template, you can use the attribute useclass for Generate Objects extending from TNode with specific properties like SQL-Java-Mappings or something like that.

5 License

GNU General Public License.

6 FAQ

6.1 Why does JavaTemplates ignore any changes in my jsp template files?

This is due to a bug in JDK1.3: The implementation of isModified in java.io.File and java.net.URL is inconsistent. This is a known bug. Please remove the scratch dir before rendering :-(

6.2 Is JavaTemplates an academic tool or is also usable in real life?

JavaTemplates is devoloped in the context of a great Webapplication based on Jakarta-Struts and Jakarta-Tomcat called Projectforge (
www.projectforge.org www http://www.projectforge.org
). The basic idea of ProjectForge was to develop a free group ware. Actually round about 200 files will be generated automatically (SQL-Scripts, Java classes (DBWrapper, logic, struts action and form classes, java server pages and also configuration files like Struts-config.xml).
The advantage of usage of JavaTemplates is the short time for the implementation of new features and the easy way to develop independent applications (e. g. independent of database).

6.3 What about XSLT?

It's your own decision to use XSLT or JavaTemplates / JSP. To render XML-Files needs more complex structures of programming language I will prefer JavaTemplates / JSP. For rendering XML to XML or documents I think XSLT is really useful.
One advantage of JavaTemplates / JSP is that you can use any Java objects you want inside your template files called from your own application (see the email example above).