Jamama Developer Guide

Blaine Simpson

Jamama Developer Guide

Blaine Simpson

Published $Date: 2004/04/14 12:45:42 $

Table of Contents

List of Tables

List of Examples

Introduction

If you notice any mistakes in this document, please email me at blaine.simpson@admc.com [mailto:blaine.simpson@admc.com?subject=dev%20Guide] so that I can correct them. I've just converted this from Texinfo to Docbook, so there are most likely many mistakes in this revision. You can also email me if you have problems with the procedures explained herein, or if you have questions, comments, suggestions or complaints.

Available formats for this document

This document is available in several formats.

You may be reading this document right now at http://jamama.sourceforge.net/dev, or in a distribution somewhere else. I hereby call the document distribution from which you are reading this, your *current distro*.

http://jamama.sourceforge.net/dev hosts the latest versions of all available formats. If you want a different format of the same *version* of the document you are reading now, then you should try your current distro. If you want the latest version, you should try http://jamama.sourceforge.net/dev.

Sometimes, distributions other than http://jamama.sourceforge.net/dev do not host all available formats. So, if you can't access the format that you want in your current distro, you have no choice but to use the newest version at http://jamama.sourceforge.net/dev.

Table 1. Alternate formats of this document

format	**your distro**	**at http://jamama.sourceforge.net/dev**
Chunked HTML	index.html	http://jamama.sourceforge.net/dev/index.html
All-in-one HTML	dev.html	http://jamama.sourceforge.net/dev/dev.html
PDF	dev.pdf	http://jamama.sourceforge.net/dev/dev.pdf

Chapter 1. Building

If you don't know what Jamama is, see the Jamama Design Document [../design/index.html].

1. Obtain and install JDK 1.4.x.

2. Obtain and install Sun's Web Service Developer Pack, JWSDP.

3. Obtain and install Ant version 1.5.3 or greater and make sure that it is in your search path.

4. Obtain the Ant contrib jar and put it into the `lib` subdirectory under your base Ant installation diretory. (If the contrib jar file does not have a version number in the file name, I recommend that you rename it with the version number). Ant Contrib can be downloaded from http://sourceforge.net/project/showfiles.php?group_id=36177.

5. Export the variable **JAVA_HOME** to the base directory of your JDK. Ant requires this, regardless of whether javac is in your search path.

6. Obtain source code using CVS or by downloading the source zip at https://sourceforge.net/project/showfiles.php?group_id=98416 and unzipping it.

   If you use CVS, the distribution will expand into a directory under your current directory named *jamama*. If you use unzip, the distribution will expand into a directory under your current directory named *jamama* plus version number. (If you got the zip from somewhere other than the SourceForge Files page, then the version number may end with "+". If so, then the code does not correspond directly to a Jamama release but has some development module modifications on to of the version without the +). The root directory "jamama" or "jamama" and version numer is referred to as **JAMAMA_HOME** in the Jamama documentation (including this document).

7. If you don't want to set up your own runtime logging system, then copy the sample files in **JAMAMA_HOME**/localres to the same filenames without the `.sample` suffixes.

8. Copy the sample file in **JAMAMA_HOME** /config to the same filename without the `.sample` suffix.

9. Look at the top of **JAMAMA_HOME**/build.xml. If the value of *jwsdp.home* is not set to the directory that you installed JWSDP to, then make a `build.properties` file in the same directory and set the value of *jwsdp.home* like this.

   ```
   jwsdp.home:  /path/to/where/you/installed/jwsdp
   ```

10. Run `ant -projecthelp` in the **JAMAMA_HOME** directory. This shows you all the stuff that you can do using ant. You always run ant from the **JAMAMA_HOME** directory.

    N.b. that interactive mode of Jamama does not work when it is run through Ant. This is because Ant does something with stdin/stdout so that your input never makes it to the application. Because of this, you have to shut down Jamama by using a JMX interface (like the Html adaptor) or just Ctrl-C at the command-line.

11. When you run Ant with the Jamama build file, if you ever get any error messages or warnings telling you that some variable is not set, then set it in a properties file in the same directory named *build.properties*. (If it's a one-time setting... or to override a setting in build.properties or build.xml, you can set properties on the Ant command line like

```
ant -Dpropname=propvalue -Dothername=otherval target1...
```

To add things to your classpath (like a log4j jar file), set the property 'local.runtimeclasspath' in your build.properties file. It takes a colon/subcolon delimited path. You don't need to add the stuff in **JAMAMA_HOME**/lib, but you do need to add **JAMAMA_HOME**/localres if you are using resources from that directory.

12. See the Jamama User's Guide [../user/index.html] for instructions on how to run Jamama without Ant (which you will want to do if you want to use the interactive command-line as opposed to working with the HTML, XML file, or other interfaces).

13. Also see the Jamama User's Guide [../user/index.html] for how to configure Jamama, including how to set up logging.

Chapter 2. Methodology and Conventions

Methodology

XP [http://www.extremeprogramming.org], aka Extreme Programming [http://www.extremeprogramming.org].

Coding Conventions

Please use good descriptions when adding files or checking in files to CVS (i.e. `cvs... add` and `cvs... commit`).

Unit test.

Sun's Java Coding Conventions [http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html].

(You are encouraged to berate Blaine when he skips the braces around blocks of a single statement-- at least he does always indent).

Chapter 3. API Specs

I trust that you already have access to your JDK API.

- The Jamama Javadoc API [../api/index.html]
- Java Utilities API [http://admc.com/java/util/doc/api/]
- JavaMail API [http://java.sun.com/products/javamail/javadocs/index.html]
- Ant v 1.5.4 [http://ant.apache.org/manual/index.html]
- Ant Control v 0.5 [http://ant-contrib.sourceforge.net/tasks/index.html]

Chapter 4. CVS

The Jamama source code repository has moved to Sourceforge. Watch mode is on, so you need to use the command `cvs edit` to publicize your intention to change source code.

Howto

See the Resources [http://admc.com/blaine/howtos/cvs/cvs-all.html#Resources] and Traditional Client [http://admc.com/blaine/howtos/cvs/cvs-all.html#Traditional] sections of my CVS HOWTO [http://admc.com/blaine/howtos/cvs/cvs-all.html].

Watch Mode

CVS watch mode is on for the following reasons.

- You can't accidentally modify a file, since you must run `cvs edit` before you can modify.
- You can find out who is currently editing files by running `cvs editors`
- If you have a particular interest in any set of resources, you can receive notification whenever any of those items are edited, unedit, or committed.

Watching is **not** strict locking. Five people can be editing a file at the same time, and CVS will handle those merges in the same way it always does. The watch feature just makes files read-only until you run `cvs edit` and lets people see everybody who is editing resources.

To release an edit, just commit your change or run `cvs unedit`.

Before you run these commands, do yourself a favor and make a .cvsrc file containing

```
cvs -z6 -q
```

This will use compression to speed up CVS transfers, and will turn off the Windows-like verbose messages that CVS is fond of. Users of third party CVS clients or wrappers will have to find out how to set these default for their client.

Unless you enjoy typing your password over and over, create a ssh key and upload it to the Sourceforge CVS server under your Account Options page at Sourceforge.

Typical CVS development procedure

Check out everything from the tip of the jamama-dev CVS module.

Example 4.1. CVS checkout of the Jamama development CVS module

```
$ cvs -d :ext:cvs.sourceforge.net:/cvsroot/jamama co
$ cd jamama-dev
```

Do your work

Edit files, unedit files, update, run ant commands to your heart's content. Here is a very small subset of some stuff you could do.

Example 4.2. Working with resources with watch mode on.

```
$ cd src/com/admc/jamama
$ cvs edit AFile.java
$ cvs update
$ cd ../../../../docsrc/adoc
$ cvs unedit docfile.texi
$ cd ../..
$ ant update
$ ant
$ ant doc
```

When you are finished, commit your changes.

Example 4.3. Preparing for committing.

```
$ cd /path/to/jamama-dev
$ cvs update -d  # You will be notified of conflicts
```

Edit conflicting files (they are labelled with `C` in the update listing), and make sure the code still builds. Repeat until the cvs update command shows no `C`'s. Continue once all conflicts are resolved...

Commit all of your additions

```
$ cvs commit -m 'Your comment'
```

You can, of course, commit resources individually or by subdirectory branches. You will want to check in files individually if your changes warrent different log messages for different resources-- *Use meaningful log messages!*.

Chapter 5. References

Java Documentation

- See the API and Conventions sections for references to those docs.
- Jamama Design Document ../design/index.html
- Jamama User's Guide ../user/index.html

Relevant Articles

- Article *Test email components in your software*. He makes a Java SMTP server for test purposes, which ignores the mail content, and does no relaying. http://www.javaworld.com/javaworld/jw-08-2003/jw-0829-smtp.html
- JavaMail. Not that good. 2nd page on POP. 3rd not about mail at all. http://javaworld.com/javaworld/jw-10-2001/jw-1026-javamail.html

Internet RFCs

RFC 1869 [http://www.faqs.org/rfcs/rfc1869.html]	ESMTP. (Trivial).
RFC 2821 [http://www.faqs.org/rfcs/rfc2821.html]	SMTP.
RFC 821 [http://www.faqs.org/rfcs/rfc821.html]	Old SMTP.
RFC 2822 [http://www.faqs.org/rfcs/rfc2822.html]	Internet Message Format
RFC 822 [http://www.faqs.org/rfcs/rfc822.html]	Old Internet Message format
RFC 2123 [http://www.faqs.org/rfcs/rfc2123.html]	Also see app/support
RFC 1985 [http://www.faqs.org/rfcs/rfc1985.html]	ETRN
RFC 1870 [http://www.faqs.org/rfcs/rfc1870.html]	SIZE
RFC 1653 [http://www.faqs.org/rfcs/rfc1653.html]	SIZE
RFC 1427 [http://www.faqs.org/rfcs/rfc1427.html]	SIZE
RFC 3030 [http://www.faqs.org/rfcs/rfc3030.html]	Messages, SMTP Service Extensions for Transmission of Large and Binary MIME
RFC 1891 [http://www.faqs.org/rfcs/rfc1891.html]	DSN (delivery status notifications)
RFC 1894 [http://www.faqs.org/rfcs/rfc1894.html]	DSN (delivery status notifications)

/rfc2476.html]	Message Submission
RFC 2298 [http://www.faqs.org/rfcs/rfc2298.html]	maybe Message Submission
RFC 1047 [http://www.faqs.org/rfcs/rfc1047.html]	duplicate msgs
RFC 2045 [http://www.faqs.org/rfcs/rfc2045.html]	MIME (part 1)
RFC 2046 [http://www.faqs.org/rfcs/rfc2046.html]	MIME (part 2)
RFC 2047 [http://www.faqs.org/rfcs/rfc2047.html]	MIME (part 3)
RFC 2048 [http://www.faqs.org/rfcs/rfc2048.html]	MIM (part 4)
RFC 1652 [http://www.faqs.org/rfcs/rfc1652.html]	SMTP Service Extension for 8bit-MIMEtransport
RFC 1426 [http://www.faqs.org/rfcs/rfc1426.html]	SMTP Service Extension for 8bit-MIMEtransport
RFC 1830 [http://www.faqs.org/rfcs/rfc1830.html]	CHUNKING. Large and Binary MIME
RFC 2076 [http://www.faqs.org/rfcs/rfc2076.html]	HEADERS
RFC 2197 [http://www.faqs.org/rfcs/rfc2197.html]	Pipelining
RFC 2920 [http://www.faqs.org/rfcs/rfc2920.html]	Pipelining
RFC 3463 [http://www.faqs.org/rfcs/rfc3463.html]	Enhanced Mail System Status Codes
RFC 3463 [http://www.faqs.org/rfcs/rfc3463.html]	Enhanced Mail System Status Codes
RFC 1893 [http://www.faqs.org/rfcs/rfc1893.html]	Enhanced Mail System Status Codes
RFC 1985 [http://www.faqs.org/rfcs/rfc1985.html]	Enhanced Mail System Status Codes (These are the numeric codes after the normal SMTP status codes.)
RFC 2034 [http://www.faqs.org/rfcs/rfc2034.html]	Email DNS: (part 1),
RFC 974 [http://www.faqs.org/rfcs/rfc974.html]	Common Mailbox Names (part 1),
RFC 2142 [http://www.faqs.org/rfcs/rfc2142.html]	Authentication
RFC 2554 [http://www.faqs.org/rfcs/rfc2554.html]	

Chapter 6. Technologies

Commons Logging [http://jakarta.apache.org/commons/logging/userguide.html]	JMX/JMXR [http://admc.com/blaine/howtos/jmx/]. Dynamically add and remove objects to the running JVM + Remote Management. You only need to understand Agent/JMX Client implementation if you will be working with the main Jamama class (which is a JMX Agent). See my JMX Howto at http://admc.com/blaine/howtos/jmx/ You should understand JMX notifications, because we will be implementing that in Jamama soon. You do need to understand the AbstractDynamicMBean utility which basically creates and manages a DynamicMBean for you. AbstractDynamicMBeans are a product of MX4J, and I cover them in my Howto.
JAXB [http://java.sun.com/webservices/docs/1.2/tutorial/doc/JAXBWorks.html#wp100322]	Java Objects <---> XML for configuration persistence and restoration. I intend to write a HOWTO for Jaxb. When I do that, I'll put a link to the HOWTO here. The best documentation source for JAXB that I know of is at http://java.sun.com/webservices/docs/1.2/tutorial/doc/JAXBWorks.html#wp100322}
Ant and Ant-contrib	See the APIs above.
JUnit [http://admc.com/blaine/howtos/junit/]	Unit testing. See my JUnit Howto at http://admc.com/blaine/howtos/junit/
Hsqldb Database [http://hsqldb.sourceforge.net/]	A database very well suited for embedding in this app. Small, fast, embeddable, efficiently handles data or varying sizes (including veyr large sizes). As of 11/28/2003, I don't have the database implemented at all yet. I'll post the Hsqldb Beginner's Guide once I have it updated (it will also be distributed with Hsqldb v. 1.7.2 when it is released).

Chapter 7. Howto

Add a new component

1. Adding an existing component If you want to just use a new compnent, then see the User's Guide. This section talks about implementing a new Jamama component class.

2. Decide what interface(s) your new component will implement It must implement *Configurable*. Decide if it will also implement any (or any combination of) the following Configurable subinterfaces.

   - Server
   - ThirdPartyConfiguror
   - Router
   - Repository

3. Add a new <xsd:complexType> block under the 'instances' element underneath all of the other instances. It is usually easiest to find another <xsd:complexType> under 'instances', copy and paste it and edit the copy.

4. Validate the schema (if you have a schema validator) Run a schema validator against the .xsd file, or (better) make a Jamama xml config file that would instantiate one of your new objects, sure the schemaLocation points to your modified .xsd file. and validate the XML with an XML validator that validates schemas.

   Put references for goot W3C Schema validators.

   Fix the schema until it validates successfully.

5. Comple the jaxb source Run `ant compile-jaxb` Fix the schema until xjc is satisfied with it and can generate all the Java code.

6. Implement a class that implements the interfaces you decide upon. Work from the API located at **JAMAMA_HOME**/doc/api. If it's not there, then built it like `ant api`. You will need to cast the Object argument that you get in your configure() method into the new class that you will find in the API under the com.admc.jamama.dtd package. The methods for that class will show exactly how to set your startup state in configure(). Perform the inverse operations in your getConfig() method and return the same configuration type object (the JVM will cast it down to an Object).

   If there is an suitable adapter available, subclass the adapter and save yourself some work.

7. Update the Jamama class. Add one element to the private field Jamama.supportedClassArray. The element consists of the JAXB class and the Configurable class that you are adding. The former can

be found in the Jaxb API and will be of the form com.admc.jamama.dtd.ConfigType.InstancesType.X.

If you added your new element underneath the existing instance elements, then I think that the method name of cfg.getInstances().X() should not have changed. Until I sure of this, do check getter java.util.List method in the API for com.admc.cjamama.dtd.cnfigType.InstancesType. (If it has changed, then you willhave to change where that getter method is used in Jamama.java).

8. Get everything to build and run successfully.