How to build JVoiceXML 0.7.5.EA

Version 1.9.6
Date November 30, 2011

Dr. Dirk Schnelle-Walka
dirk.schnelle@jvoicexml.org

Contents

Abstract

This documents describes the steps you have to perform, when you want to build JVoiceXML or develop code for the JVoiceXML project. It gives information about the requirements of your development environment and our coding conventions.

1 Introduction

JVoiceXML is a free VoiceXML [16] implementation written in the JAVA programming language. It offers a library for easy VoiceXML document creation and a VoiceXML interpreter to process VoiceXML documents using JAVA standard APIs such as JSAPI [11] and JTAPI [11].

VoiceXML is hosted at SourceForge [10] as an open source project. You find everything that is related to this project under `http://sourceforge.net/projects/jvoicexml/`.

This document refers to UNIX and Windows systems. JVoiceXML will work any other operating systems that support Java 6, too.

Nobody is perfect, so you may find some errors or small things to correct. Please let me know if you think you found something that should be written differently.

2 Copyright

JVoiceXML uses the GNU library general public license [6]. This is mentioned in all our source files as a unique header, see section 10. You can find a copy in the file COPYING in the ${JVOICEXML_HOME} directory. This means that you are allowed to use JVoiceXML library in your commercial programs. If you make some nice enhancements it would be great, if you could send us your modifications so that we can make it available to the public.

JVoiceXML is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

JVoiceXML is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.

You should have received a copy of the GNU Library General Public License along with this library; if not, write to the Free Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

3 Download the Source Code

3.1 Download the source archive

You can download a the source code for the available releases from `http://jvoicexml.sourceforge.net/downloads.htm`. The source code is part of the binary distribution and can be obtained by running the installer. This is not the recommended way. Please use the SVN repository as described in the following sections.

3.2 SVN repository

If you want to stay at the current state of development you have to use the SVN repository from Source Forge [10]. This is also the preferred way.

The SVN repository is organized as a multi-project repository with the following sub projects:

core The core of the JVoiceXML voice browser

demo Demo programs.

documentation LaTeXsources for the documentation

eclipse Eclipse server plug-in

systemtest Source for the VoiceXML compatibility test

Each subproject has its own trunk, tags, and branches folder.

This section describe the parameters to access the SVN repository using the command line. Feel free to grab the required parameters from this description and feed your favorite tool with them.

```
1 svn co \
https://jvoicexml.svn.sourceforge.net/svnroot/jvoicexml/<subproject>/trunk \
jvoicexml
```

This will check out all the projects of the `trunk` of the subproject `<subproject>` into the folder `jvoicexml`.

Since all the third party libraries are checked in this may take a while.

Although the sub-projects are organized in folders it is expected to check-out all projects into the same folder regardless of the subproject folder. This works fine, if you are using an IDE like eclipse, this is really straightforward, but may cause some problems if you are working from a command line. In the latter case remove the `.svn` folder if you checkout another subproject. Otherwise SVN will complain that there is already another one using this folder. A screenshot from a checkout in eclipse is shown in figure 1.

Note: UNIX file and directory names are case sensitive.

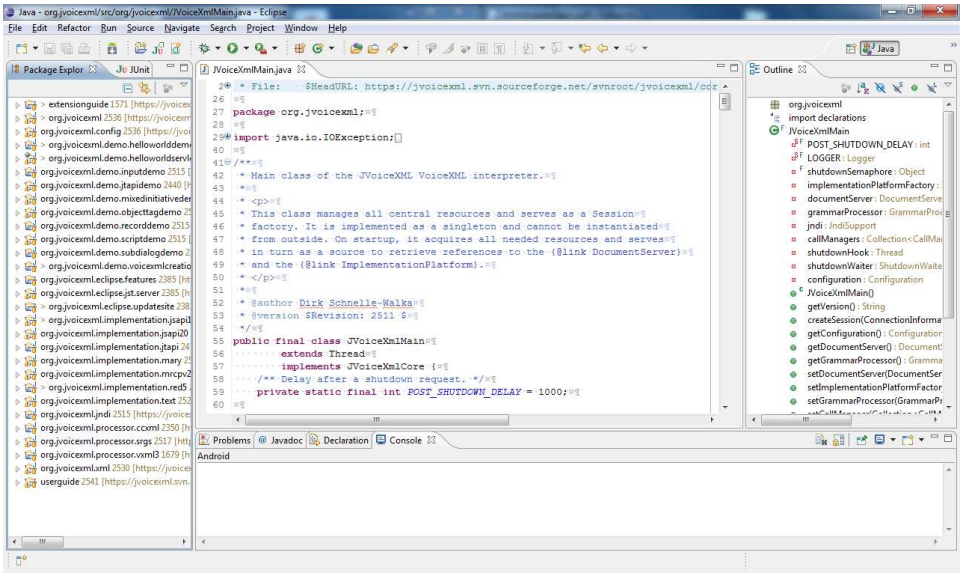

Figure 1: Eclipse project overview of JVoiceXML

4 Required Software

Since JVoiceXML is written in JAVA you will at least need a JAVA compiler, see section 4.1, an editor or preferably a JAVA IDE, see section 4.1, and ANT, see section 4.3, to build the binaries. All used third party libraries can be downloaded from the CVS, SVN or comparable repository or from their home pages. The file `doc/libraries.xhtml` contains a list of all used libraries.

4.1 IDE

JVoiceXML is being developed using eclipse 3.4, but you can use the IDE of your choice to edit the sources and compile the binaries. You can even use a simple text editor to perform this job. Nevertheless there are some restriction that you cannot work around.

Your IDE must support

- J2SE 6
- ANT 1.7

4.2 JAVA

Parts of the code of JVoiceXML are using features from the JAVA 5 API, so that you will need at least J2SE 1.6 to compile the code. You can download it for free from `http://java.sun.com/javase/`.

4.2.1 Eclipse settings

If you are not using eclipse as your favorite IDE, you may ignore this section. We are using the *Eclipse IDE for Java EE Developers* version 3.5.2 for the development of JVoiceXML. The subversion repository contains the eclipse project settings. However you are requested to configure the execution environment development for Java 6 *JavaSE-1.6* to point to a valid JDK. The use of execution environments makes the development independent of the used version of the Java 6 SDK.

Eclipse Plug-ins The development of JVoiceXML relies on some plug-ins that are not part of the IDE. You will need to install the following eclipse plug-ins via the eclipse Software Update mechanism.

Below is a list of the required update URLs.

Checkstyle `http://eclipse-cs.sourceforge.net/update` (Note that you only need the 5.x version)

Subversion `http://subclipse.tigris.org/update_1.6.x`

TeXlipse `http://texlipse.sourceforge.net` (optional if you want to edit the documentation)

4.3 ANT

JVoiceXML is being built by an ANT build file It is recommended that you use at least ANT 1.7.0. If you don't have ANT installed, you can download the current release from `http://ant.apache.org`.

The build file allows you to override the settings by using a custom properties file jvoicexml.properties in the ${JVOICEXML_HOME} directory.

Nearly all IDE's feature an ANT integration. This allows to use your favorite IDE.

4.4 Tomcat

For the system test and for some demos you will need to have a servlet container installed. We use Tomcat 5.5 for that purpose. You can download this release from `http://tomcat.apache.org`.

This is not needed if you do not use the servlet based demos or want to run the system test.

4.5 LaTeX

The documentation is being written using LaTeX. If you want to edit the documentation you will need to install a LaTeXsystem like MiKTeX `http://miktex.org` for windows or tetex on Linux systems.

This is not needed if you do not want to author the documentation or create a distribution.

5 JVoiceXML Core

The core of JVoiceXML comprises the following sub-projects:

org.jvoicexml The core of the interpreter (Required)

org.jvoicexml.config JVoiceXML configuration based on spring (Required)

org.jvoicexml.xml The JVoiceXML XML library (Required)

org.jvoicexml.jndi JNDI support for JVoiceXML (Required)

org.jvoicexml.processor.srgs SRGS processor (Required)

org.jvoicexml.implementation.jsapi10 Demo implementation platform for JSAPI 1.0 [11].

org.jvoicexml.implementation.jsapi20 Demo implementation platform for JSAPI 2.0 [8].

org.jvoicexml.implementation.jtapi Demo implementation platform for JTAPI 1.3 [12].

org.jvoicexml.implementation.mary Demo implementation platform for the Mary Open TTS system.

org.jvoicexml.implementation.marc Demo implementation platform for the MARC.

org.jvoicexml.implementation.mrcpv2 Demo implementation platform for MRCPv2.

org.jvoicexml.implementation.text Demo implementation platform for text based access.

5.1 Project `org.jvoicexml`

This project is the core of the JVoiceXML browser and the central point of building and configuring the environment.

5.1.1 Directory Structure

Having the source code in your ${JVOICEXML_HOME}/org.jvoicexml directory you find the following directory structure:

3rdparty third party libraries. This directory contains all libraries that are used by the core or by more that one subproject. Hence, it serves as a central library repository.

classes compiled binaries

config configuration setting to run the voice browser

config-props configuration templates for the ant build files

dist distribution files

etc resource files

personal-props adapted configuration from the configuration templates for the ant build files. The contents of this folder is never committed to the SVN repository except the `README.TXT`.

logging logging output of JVoiceXML

src Java source code

test JUnit tests and other tests

work Recorded audio files

3rdparty-libs.xml ANT setting for the third party libraries

build.xml main ANT build file

configuration.xml ANT build file to create a customized configuration

distribution.xml ANT build file to create the distribution

nightly.xml ANT build for the nightly build process

run.xml ANT build file to run the voice browser

test.xml ANT build file to to run the unit tests

platform-targets.xml Included ANT build file that will call the configured implementation platforms (see sections 5.1.2 and 5.1.7)

Not all of the files are present at start but are created by the ANT build file or through procedures described in this document, see section 5.1.8 or while running the interpreter.

5.1.2 Adapting the ANT Configuration

The directory `personal-props` provides a developer with the possibility to override any default properties defined by files in `config-props`. Any file that is to be substituted must be copied from `config-props` (preserving the directory structure) to that directory and can then be modified. Do not modify the settings in the `config-props` folder. This is a possible source for SVN conflicts.

The settings in the build.xml files assure that the files found in this directory have precedence over those in config-props.

5.1.3 Third Party Libraries

JVoiceXML uses some third party libraries. This section names them and tell you where you can get them. All of them are at least freeware so that it was possible to store them in the SVN repository at SourceForge. You can download them either from that location or from their home pages.

In this section, only those libraries are described, that are required to build the main project. The demos may require additional libraries.

You can also use your local copy by adjusting the settings in your custom properties file.

The settings of these libraries are stored in the file *3rdparty-libs.xml* which is imported by the main build file.

All third party libraries are located in the directory ${JVOICEXML_HOME}/3rdparty.

This directory is assigned to the property `3rdparty.dir`

Each library is located in a specific subfolder of the ${3rdparty.dir} folder. These subfolder follow this convention:

- The name of the subfolder is a combination of the library name and the version of this library, e.g. `log4j1.2.15` for the *log4j* library with the version number*1.2.15*.
- The jars of this library are located in a subfolder of this folder called *lib*
- The Javadoc documentation for the library is compressed into a zip archive and added to the subfolder.

You find a copy of the license terms for the libraries in the folder ${JVOICEXML_HOME}/etc/legal.

Library Snapshots Some of the used libraries are under active development and have not yet released a patched release. In these cases the lib

folder contains a snapshot from their repository. The presence of a snapshot is marked in the file `doc/libraries.xhtml` e.g. with the comment *(SVN snapshot from ¡date¿)*.

5.1.4 Libraries Needed to Compile JVoiceXML

log4j 1.2.15 JVoiceXML uses log4j [1] for logging. We think that log4j has some advantages over `java.util.logging` and appears to be more mature and reliable.

The output of SUN's own logging framework can be redirected to log4j via the system property

```
-Djava.util.logging.config.file=${config}/logging.properties
```

This file contains the description of a logging handler that forwards all request to log4j. Hence, both the log4j logging framework and SUN's logging framework can be configured through the log4.xml configuration file.

Rhino 1.7R2 JVoiceXML uses rhino [9] from Mozilla to enable scripting. Java 6 already comes with a rhino implementation, but since we need to build custom components we can not rely on this.

Commons pool 1.5.3 The pooling of implementation platform resources is based on commons pool.

Commons HTTP client 4.0 The HTTP client library is used to retrieve VoiceXML documents from a web server.

5.1.5 Libraries Needed at Run-time

Besides the libraries named in section 5.1.4 the following libraries are needed to run JVoiceXML.

Commons Logging 1.1.1 This library is required by the other commons libraries.

Commons Codec 1.4 This library is required by the HTTP client to add attachments to the POST and GET requests.

5.1.6 Other Libraries

Besides the libraries named above there are also other libraries in the 3rd-party folder. These libraries are used by more than one other implementation platforms and are kept here as a general pool. Libraries that are used only by one single implementation platform are located in the implementation platform's 3rdparty library directory.

5.1.7 Creating a Configuration

Currently there are seven implementations for the implementation platform that reside in co-projects, refer to section 3.2.

Section 5.1.2 describes how to prepare your personal settings. If you want to use one of them, you have to adapt the corresponding setting in your copy of `platforms.xml` in the folder`personal-props` by simply uncommenting it. An example to enable the JSAPI 1.0 platform is shown below:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<platforms>
  <!--
    Uncommment the platforms that you want to use or add your own platform.
  -->
  <platform name="org.jvoicexml.implementation.jsapi10">
    <property name="jvxml.jsapi10.talkingJava" value ="false" />
    <property name="jvxml.jsapi10.talkingJava.path"
        value ="../org.jvoicexml.implementation.jsapi10/3rdparty/talkingjava/
            lib" />
  </platform>
  <!-- platform name="org.jvoicexml.implementation.jsapi20">
    <property name="jvxml.jsapi20.sapi" value ="false" />
    <property name="jvxml.jsapi20.mac" value ="false" />
  </platform-->
  <!-- platform name="org.jvoicexml.implementation.jtapi"/-->
  <!-- platform name="org.jvoicexml.implementation.mary"/-->
  <!-- platform name="org.jvoicexml.implementation.marc"/-->
  <!-- platform name="org.jvoicexml.implementation.mrcpv2"/-->
  <!-- platform name="org.jvoicexml.implementation.red5"/-->
  <!-- platform name="org.jvoicexml.implementation.text"/-->
</platforms>
```

This will enable the JSAPI 1.0 platform to work with Sphinx 4 and FreeTTS.

This setting is also used if you call ant for the main build file as described in the following section. Note that you need at least one implementation platform to use the voice browser.

Once you changed the settings call

```
ant -buildfile configuration.xml
```

to create your adapted configuration settings. In addition you also get a customized ant build file `run.xml` as well as the included `platform-targets.xml` to start and stop the interpreter using the libraries from the co-projects.

Note that it is required to create a configuration after you changed the settings in the `ant.properties` file.

The voice browser can be started by

```
ant -f run.xml run
```

and stopped by

```
ant -f run.xml shutdown
```

Please do not stop the voice browser by CTRL-C or similar. It may occur that some resources are not closed properly and may cause conflicts with the next start of JVoiceXML.

The created `run.xml` also contains a target to enable remote debugging. It can be called by

```
ant -f run.xml debug
```

The code will be compiled using your current configuration and the start of the voice browser will be delayed until you connect with a remote debugger. The used server port is 12367. This allows to set breakpoints using your IDE and debug JVoiceXML.

5.1.8 The Main Build File

This section explains the most important targets of the main build file `build.xml`. All others buildfiles from the implementation platforms are called from this central point. Just start ant without any target specified if you want to build everything. Call

```
ant -projecthelp
```

to get an overview of the targets and their purpose.

Some of the targets use third party extensions to ANT. These extensions are not described in this document, but are part of the 3rdparty folder. In contrast to the libraries to link with, these libraries are not defined in the file `3rdparty-libs.xml` but in the corresponding target of this build file. Each features a preceding target that checks if the library is required. If the library is not found, the target will not be called.

clean Delete all compiled class files in the directory *classes* and the jars that are created by the *jar* target in the directory *dist*.

compile Compile all JAVA files in the directory *src* into the directory *classes*.

jar This target depends on the target *compile* and creates the jar files of your distribution in the directory *dist*. If successful, you will find at minimum the following jar archives:

- jvxml.jar This jar file contains the core of JVoiceXML.
- jvxml-xml.jar This jar contains all files that are required to create and parse VoiceXML documents. This file is being build from the project `org.jvoicexml.xml` which is described in section 5.3.
- jvxml-client.jar This jar contains the files needed to build a client.
- jvxml-jndi.jar JNDI support for JVoiceXML

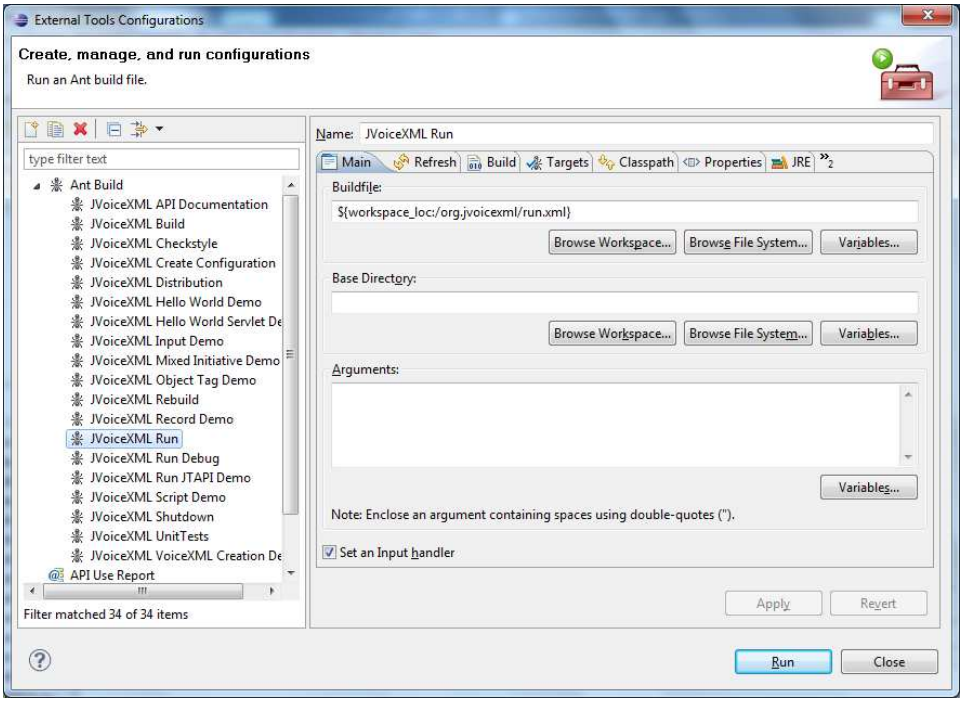

Figure 2: Eclipse External Tools Configuration for JVoiceXML

- some other jars, depending on your configuration settings

apidoc Create JAVADOC documentation from the JAVA files in the directory doc/api.

checkstyle Perform a check of the JVoiceXML coding standard as specified in section 10.

Call

```
ant <target>
```

to run ant for the the given target.

The ANT launch configurations for eclipse are part of the SVN repository. They are available as a shortcut in the external tools configuration after a first run. A screenshot is shown in figure 2.

5.2 Project `org.jvoicexml.config`

This project provides a library to configure JVoiceXML based on spring. This is not required at run time and can be replaced by custom implementations, e.g. to embed JVoiceXML. It s required to build JVoiceXML.

This project has to be build after the main project.

5.2.1 Third party libraries

Spring Framework 3.0.5 JVoiceXML uses the spring framework to address configuration issues.

5.3 Project `org.jvoicexml.xml`

This project provides an XML library to create and parse all VoiceXML related XML languages like SRGSXML, PLS and SSML.

This project has to be build before the main project.

5.4 Project `org.jvoicexml.processor.srgs`

This project provides an SRGS processor.

This project has to be build before the main project.

5.5 Project `org.jvoicexml.jndi`

This project provides JNDI support for JVoiceXML. This is not required at run time but in order to build JVoiceXML.

JNDI support offers remote access to JVoiceXML, e.g. the ability to run the shutdown script or to run the demos.

The project settings for this subproject can be adapted by the following ANT properties:

```
jvxml.jndi.repository=text
jvxml.jndi.port=1099
```

The configuration ANT script also adapts the settings in the `jndi.properties` to the configured port. Make sure to also adapt the `jndi.properties` in the demos or your application.

The repository entry must match the repository that you are using for your client. Otherwise JVoiceXML will not be able to load the needed resources of the server side.

5.6 Project `org.jvoicexml.implementation.jsapi10`

This project provides an implementation platform for JSAPI 1.0 compliant speech engines. Besides a general hook for those engines there are also extensions for FreeTTS and Sphinx 4 bases on these hooks to demonstrate the capabilities.

The project settings for this subproject can be adapted by the following properties in the `platforms.xml`

jvxml.jsapi10.talkingJava A value of `false` will result in a configuration that uses FreeTTS and Sphinx4 as speech engines. If set to `true` the Talking Java wrapper for Microsoft's speech engine will be used. In the latter case you need also to specifiy a value for `jvxml.jsapi10.talkingJava.path`

jvxml.jsapi10.talkingJava.path A path that is used to locate the DLLs that are required to use Talking Java as sthe speech engine.

5.6.1 Third party libraries

JSAPI 1.0 This implementation platform uses the Java Speech API v1.0 (JSAPI) [11] to address speech recognition and speech synthesis issues.

FreeTTS 1.2.2 JVoiceXML uses FreeTTS [5] as a demo implementation for TTS output. FreeTTS comes with two libraries: One for the general functionality and one custom for the JSAPI 1.0 support. The general library is kept in the core project's 3rdparty library directory and the JSAPI library can be found in this project's 3rdparty library directory.

Note that the used library is a snapshot from the FreeTTS SVN repository.

CMU sphinx 4 JVoiceXML uses sphinx 4 [3] from Carnegie Mellon University as a demo implementation for speech recognition.

Note that the used library is a snapshot from the sphinx SVN repository.

Talking Java Talking Java is an implementation of the JSAPI 1.0 utilizing Microsoft's SAPI4 and SAPI5 speech engines. It can be used as a replacement for FreeTTS and CMU sphinx4. Note that the delivered version is only free for personal use. If you intend to use it in a commercial corporate or institutional context, you will have to buy a license. Please refer to `http://www.cloudgarden.com`.

5.7 Project `org.jvoicexml.implementation.jsapi20`

This project contains a first draft of an implementation platform to enable JSAPI 2.0 support for JVoiceXML.

This implementation platform makes use of the reference implementation (RI) that can be obtained from Conversay web page conversations [4]. Note that this is not the final release but the release candidate. It will not run on embedded devices since it makes use of Conversay's speech engine that is only compatible with the Windows operating system.

5.7.1 Third party libraries

JSAPI 2.0 The JSAPI 2.0 specification is under going changes. JSAPI 2.0 is being developed within the Java Community Process (JCP) [7] under JSR-113 [8].

jsr113-jsebase This open source library library provides a framework to ease the development of JSAPI 2.0 compliant speech engines. The project also provides custom extensions for FreeTTS and sphinx. The project URL is `http://jsapi.sourceforge.net`.

Note that the used library is a snapshot from the jsapi SVN repository.

jlibrtp 0.2 jlibrtp is used to enable streaming of audio data via the RTP protocol.This is being used to stream the audio coming from the JSAPI 2.0 compliant engines to the client. The project URL is `http://jlibrtp.sourceforge.net`.

Note that the used library is a snapshot from the jlibrtp SVN repository.

FreeTTS 1.2.2 The JSAPI 2.0 implementation platform uses the general library from FreeTTS [5] as a demo implementation for TTS output.

Note that the used library is a snapshot from the FreeTTS SVN repository.

CMU sphinx 4 The JSAPI 2.0 implementation platform uses sphinx 4 [3] from Carnegie Mellon University as a demo implementation for speech recognition.

5.8 Project `org.jvoicexml.implementation.jtapi`

This implementation platform provides telephony functionality based on JTAPI to JVoiceXML.This implementation platform should be used together with other implementation platforms like the JSAPI 2.0 implementation platform since it does not have any speech functionality.

The project settings for this subproject can be adapted by the following ANT properties:

```
jtapi.sip.providername=net.sourceforge.gjtapi.raw.mjsip.MjSipProvider
jtapi.sip.address=127.0.0.1:4242
jtapi.sip.terminal = sip:jvoicexml@127.0.0.1:4242
jtapi.sip.port=4242
jtapi.sip.inputType=jsapi20
jtapi.sip.outputType=jsapi20
```

5.8.1 Third party libraries

JTAPI 1.3 JTAPI is used to address telephony issues.

GJTAPI 1.9RC2 GJTAPI is used to simply the use of the JTAPI layer. Besides the core library we use the mjsip provider from this project.

Note that the used library is a modified version. Modifications include

- bug fixes of the SIP provider

mjsip MjSIP provides an easy way to have SIP functionality. Unfortunately, this project uses the GPL license which makes is unusable in commercial applications. It should be replaced by JainSIP.

5.9 Project `org.jvoicexml.implementation.marc`

This implementation integrates MARC system into JVoiceXML. This platform only provides the output part and is designed to be used with other implementation platforms like JSAPI 1.0, see section 5.6.

Note that you will need a MARC instance running if you want to use this demo. MARC can be downloaded from `http://marc.limsi.fr/`.

This implementation platform will also for the integration of MARC BML commands into your VoiceXML files.

5.10 Project `org.jvoicexml.implementation.mary`

This implementation integrates the Mary TTS system into JVoiceXML. It is designed to be used with other implementation platforms like JSAPI 1.0, see section 5.6.

Note that you will need a Mary 4.0.0 TTS server running if you want to use this demo. Mary has to be configured as a socket server, therefore edit and file `MARY_HOME/conf/marybase.config` and change the following line

```
# Type of server? (socket/http/commandline)
server = http
```

to

```
# Type of server? (socket/http/commandline)
server = socket
```

5.10.1 Third party libraries

Mary client The Mary client is shipped with the Mary TTS systems and facilitates the communication of clients like JVoiceXML with the Mary TTS server.

5.11 Project `org.jvoicexml.implementation.mrcpv2`

This implementation platform aims at MRCPv2 support for JVoiceXML. It is in the starting phase and currently it is not usable.

This implementation platform requires the presence of an MRCPv2 server. An open source solution based on FreeTTS and sphinx 4 can be downloaded from `http://cairo.sourceforge.net`.

To make use of this implementation platform, JVoiceXML must be called with a SIP phone. We used the xlite phone which can be downloaded for free from `http://www.counterpath.com`.

The project settings for this subproject can be adapted by the following ANT properties:

```
mrcpv2.sip.address=127.0.0.1:4242
mrcpv2.sip.port=4242
mrcpv2.sip.cairo.sip.address=sip:cairo@speechforge.org
mrcpv2.sip.cairo.sip.host=127.0.0.1
mrcpv2.sip.cairo.sip.port=5050
```

5.11.1 Third party libraries

mrcp4j Mrcp4j provides the core functionality for the MRCPv2 integration.

Cairo Cairo is required as a means for communication with MRCPv2 resources.

5.12 Project `org.jvoicexml.implementation.text`

The text based implementation platform provides a text based input and output. This means that you receive the system output as Java strings and you are able to provide input via Java strings.

5.12.1 Third party libraries

No special libraries.

6 Demo Programs

The demo programs give a short impression how the JVoiceXML voice browser can be used.

The demos comprise the following sub-projects:

org.jvoicexml,demo.helloworlddemo The venerable hello world in VoiceXML

org.jvoicexml.demo.helloworldservletdemo The same but as a servlet

org.jvoicexml.demo.inputdemo A small cinema application

org.jvoicexml.demo.jtapidemo SIP based access to JVoiceXML calling the hello world servlet demo (not functional)

org.jvoicexml.demo.mixedinitiativedemo Demo to show how the mixed initiative capabilities.

org.jvoicexml.demo.objecttagdemo Demo to show how the object tag can be used.

org.jvoicexml.demo.recorddemo Demo for the record tag

org.jvoicexml.demo.scriptdemo Demo for scripting functionality with a weird dialog flow

org.jvoicexml.demo.subdialogdemo Demo to show a subdialog

org.jvoicexml.demo.voicexmlcreationdemo Demo for the VoiceXML XML library

7 Eclipse Plug in

The eclipse plug-in allows to run JVoiceXML as a server from within the eclipse IDE.

org.jvoicexml.eclipse.jst.server Eclipse plug-in to start the JVoiceXML from the eclipse server view.

org.jvoicexml.eclipse.features Eclipse features for the JVoiceXML plug-in.

org.jvoicexml.eclipse.updatesite Eclipse update site for the JVoiceXML plug-in

8 System test

The system test implements the W3C's conformance test for VoiceXML 2.0.

org.jvoicexml.systemtest Main system test project

org.jvoicexml.systemtest.servlet Servlet to access the W3C conformance test

In addition you will need the following sub-projects:

- org.jvoicexml
- org.jvoicexml.implementation.text

In order to run the system test you will need to have Tomcat 5.5 installed, refer to section 4.4. The servlet service needs to be configured via the following lines in your copy of `ant.properties` in the `personal-props` folder:

```
tomcat.home=MUST BE SUPLIED DIRECTORY
tomcat.startup=${tomcat.home}/bin/startup.sh
tomcat.shutdown=${tomcat.home}/bin/shutdown.sh
servlet.lib.dir=MUST BE SUPPLIED DIRECTORY
servlet.include=servlet-api.jar
```

In order to run the test switch to the `org.jvoicexml.systemtest` directory and call

```
ant -f night.xml run
```

The first time you call it the test suite is obtained from the W3C. If you are behind a firewall you will have to configure the proxy settings by

```
proxy.host=
proxy.port=
proxy.user=
proxy.pass
```

in the same configuration file.

9 Documentation

JVoiceXML documentation as LaTeXfiles.

The documentation comprise the following sub-projects:

extensionguide The extension guide for JVoiceXML

howtobuild The how to build JVoiceXML documentation (this document)

htdocs The HTML web pages

userguide A tutorial to start working with JVoiceXML

10 Code Conventions

We follow the JAVA code conventions [14] for our code. All methods and member variables must be commented using JAVADOC [13].

In addition we use a custom [language=Java]@todo JAVADOC tag or a `TODO` tag to mark sections that need further work.

Example:

```
/** @todo Implement the untreated case XYZ */
```

We use checkstyle [2] to check our coding conventions. All developers are requested to execute the *checkstyle* target of our ANT buildfile, see section 5.1.8. There are plug-ins for some IDEs, which you can use if you want to. The `checkstyle.xml` can be found in the folder `etc` and is called `jvoicexml-checks.xml`.

Developers are requested to use the [language=Java]@inheritDoc JAVADOC tag in favor of a [language=Java]@see reference for inherited documentation. Pros of the [language=Java]@inheritDoc tag, taken from [15], are

- satisfies checkstyle requirements for JAVADOC,
- no references to go stale,
- additional doc specific to this implementation will appear in JAVADOC and
- inherited doc in JAVADOC.

All source files must contain the following header about the copyright, see section 2, that we use for JVoiceXML.

```
/*
 * File:    $HeadURL: $
 * Version: $LastChangedRevision: $
 * Date:    $Date: $
 * Author:  $LastChangedBy: $
 *
 * JVoiceXML - A free VoiceXML implementation.
 *
 * Copyright (C) 2011 JVoiceXML group -
 *     http://jvoicexml.sourceforge.net
 *
 * This library is free software; you can redistribute it
 * and/or modify it under the terms of the GNU Library
 * General Public License as published by the Free Software
 * Foundation; either version 2 of the License, or (at your
 * option) any later version.
 *
 * This library is distributed in the hope that it will be
 * useful, but WITHOUT ANY WARRANTY; without even the
 * implied warranty of MERCHANTABILITY or FITNESS FOR A
 * PARTICULAR PURPOSE. See the GNU Library General Public
 * License for more details.
 *
 * You should have received a copy of the GNU Library
 * General Public License along with this library; if
 * not, write to the Free Software Foundation, Inc.,
 * 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
 *
 */
```

The keywords embarrassed by `$` are to be expanded by subversion. This means that for any `svn commit` the subversion option `svn:keywords` has to be set by

```
svn propset svn:keywords "HeadURL LastChangedRevision \
LastChangedDate LastChangedBy" <filename>
```

.

Since this has to be done for each file, you might want to have a look at the SVN's automatic property setting capability.

In order to activate it, locate the `config` file in you subversion home folder and open it for editing.

In the section `[miscellany]` set

```
enable-auto-props = yes
```

In the section `[auto-props]` set

```
*.java = svn:mime-type=text/plain;svn:eol-style=native;svn:\
keywords=HeadURL LastChangedRevision LastChangedBy \
LastChangedDate
```

The class comment has to contain the following information:

```
/**
 * Comment about the purpose of this class.
 *
 * @author <Name of the author>
 * @author <Other authors>
 *
 * @version $LastChangedRevision$
 * @since <current version number>
 */
```

Again the keyword expansion from subversion is used to expand the keyword, `$LastChangedRevision $` in this case. The name of the author and the purpose have to be replaced by their proper values.

11 Releases

This section describes the required steps to create a new release for JVoiceXML.

11.1 Creation of a Release Workspace

New releases are *never* built from the workspace to ensure that there are no changes in the workspace that are not reflected in the SVN repository. Therefore change to an empty directory and copy the following files from the `org.jvoicexml` project to this new folder:

1. nightly.xml
2. all jars from the `3rdparty/svnant/lib` folder

Open a command prompt at this directory and call

```
ant -f nightly.xml checkout
```

The next step is to adjust you local settings. This can easily be done by copying the contents of the personal-props directory of your development copy of JVoiceXML to the personal-props directory of the new copy.

11.2 Unit Tests

There are many unit tests based on JUnit to ensure the correctness of the classes written for JVoiceXML. Some tests need customized settings, e.g. to start the Mary TTS server. These settings are maintained in the file `config-props/test.properties` and can be overridden with a personalized copy in the folder `personal-props`.

The unit tests are started from the core project `org.jvoicexml`. It is the entry point of the unit tests of the various subproject. Therefore, change to the core directory of the release copy and call

```
cd org.jvoicexml
ant -f nightly.xml unittests
```

This will take some time. The results of the unit tests of all sub-projects can be viewed using your favorite web browser by opening `org.jvoicexml/dist/unittests/html/index` after the tests are finished. There must not be any errors or failures in the summary. Note that some tests require an active Internet connection.

If you detect any errors correct identify the cause for it, correct them yourself or contact the responsible developer of the module. After the errors have been fixed, restart from the beginning to make a new release.

11.3 Create the Distribution

After the unit test ran successfully, the binary distribution can be built. The ANT script `distribution.xml` contains the needed targets. Update the version number in the file `config-props/ant.properties` and comment the version number in the copy in the folder `personal-props`. The JVoiceXML version number consists of

```
<major-version>.<minor-version>.<bug-fix-level>.<EA|GA>
```

Early **A**ccess (EA) versions are intended for intermediate versions while **G**eneral **A**vailability (GA) versions are the main versions that are available for download via the SourceForge file downloads.

Call

```
ant -f distribution.xml
```

to create the release. Make sure that the Javadoc output does not show any errors. In case there are errors, correct them and repeat the creation of the distributable.

All intermediate files are stored in the folder `dist/version`. The distributable can be found in the folder of the core project after the executing the ANT target.

Commit the ANT properties file with the new version number.

11.4 Test the Installation

Run the installer to install the created distribution.

```
java -jar jvxml-install-<version>.jar
```

Select all available packages and test the installation. Copy the jsapi2.jar to the lib folder of the installed version. Start the voice browser and check if there are any errors in the log when JVoiceXML starts.

Uninstall JVoiceXML and reinstall it with the JSAPI 1.0 implementation platform only and the demos. Start JVoiceXML and check if all demos are working properly. It should not be necessary to restart JVoiceXML while the demos are tested.

If any error occurs while testing, locate the cause of the error, fix it yourself or identify the person who is responsible for the module and restart the release process from the beginning.

11.5 Tag the Release

The next step serves to tag the code of the release version. Therefore, you will have to supply values for the properties

```
svn.username=
svn.password=
```

in the ANT properties in the folder `personal-props`.

Call

```
ant -f distribution.xml tag
```

to create copies of the source code in the SVN folder

```
.../ <subproject>/tags/<version>
```

11.6 Upload the Distribution

The upload comprises two parts:

1. Javadoc upload
2. Release upload

Uploading requires the following properties in your personal copy of the ANT properties:

```
nightly.sf.user=
nightly.sf.password=
```

In order to upload the Javadoc, you first have to create the destination folder at SourceForge. The name of the folder is

```
/home/groups/j/jv/jvoicexml/htdocs/api/<version>
```

Make sure that the group can write the new folder and that other can access it. The generated documentation can be copied by executing

```
ant -f distribution.xml uploadJavadoc
```

The release is uploaded by

```
ant -f distribution.xml upload
```

11.7 Update of the Website

The web site needs some adaption to point to the new release when users click on the download link. The download links can be found at the page `downloads.html` in the project `htdocs`. Update the link to the new release and commit the changes.

11.8 Prepare the Further Development

Prepare the next version by incrementing th version number in the ANT properties file in the folder `config-props` and committing that file.

References

[1] Apache. log4j. `http://logging.apache.org/log4j`.

[2] checkstyle. `http://checkstyle.sourceforge.net`.

[3] CMU Shinx 4. `http://cmusphinx.sourceforge.net`.

[4] Conversay. Conversations. `http://www.conversations.com`.

[5] FreeTTS. `http://freetts.sourceforge.net`.

[6] GNU. GNU library general public license. `http://www.opensource.org/licenses/lgpl-license.php`.

[7] JCP. Java Community Process. `http://www.jcp.org`.

[8] JCP. JSR 113 Java Speech API 2.0. `http://www.jcp.org/en/jsr/detail?id=113`.

[9] Rhino: JavaScript for Java. `http://www.mozilla.org/rhino/`.

[10] SourceForge.net. `http://sourceforge.net`.

[11] SUN. Java Speech API 1.0 (JSAPI). `http://java.sun.com/products/java-media/speech/forDevelopers/jsapi-doc/index.html`.

[12] SUN. Java Telephony API (JTAPI). `http://java.sun.com/products/jtapi/`.

[13] SUN. Javadoc guidelines. `http://java.sun.com/j2se/javadoc/writingdoccomments/index.html`.

[14] SUN. SUN Code Conventions. `http://java.sun.com/docs/codeconv/`.

[15] James Tauber. Inheriting Doc in Javadoc and Eclips. `http://www.jtauber.com/blog/2004/05/11/inheriting_doc_in_javadoc_and_eclipse`.

[16] W3C. Voice Extensible Markup Language (VoiceXML) Version 2.0. `http://www.w3.org/TR/voicexml20/`, March 2004.