

# JVoiceXML 0.7.1 User Guide

Version 0.7.1.1  
Date August 6, 2009

---

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

## Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
<b>2</b>	<b>Copyright</b>	<b>4</b>
<b>3</b>	<b>Architectural Overview</b>	<b>5</b>
<b>4</b>	<b>Required Software</b>	<b>5</b>
4.1	IDE . . . . .	6
4.2	JAVA . . . . .	6
4.3	ANT . . . . .	6
4.4	Tomcat . . . . .	6
<b>5</b>	<b>Installation</b>	<b>6</b>
5.1	Implementation Platform Configuration . . . . .	7
5.1.1	Classloader repositories . . . . .	7
5.2	JSAPI 1.0 implementation platform . . . . .	8
5.3	JSAPI 2.0 implementation platform . . . . .	8
5.4	JTAPI implementation platform . . . . .	8
5.5	MRCPv2 implementation platform . . . . .	8
5.6	Text implementation platform . . . . .	9
<b>6</b>	<b>Starting the Voice Browser</b>	<b>9</b>
6.1	Linux . . . . .	9
6.2	Windows . . . . .	9
<b>7</b>	<b>Shutdown of the Voice Browser</b>	<b>10</b>
7.1	Linux . . . . .	10
7.2	Windows . . . . .	10
<b>8</b>	<b>Running the Demos</b>	<b>10</b>
<b>9</b>	<b>A first TTS example</b>	<b>11</b>
9.1	Creating the VoiceXML file . . . . .	11
9.2	Writing the Client . . . . .	12
9.3	Starting the Client . . . . .	14
<b>10</b>	<b>Creating VoiceXML using the Tag Library</b>	<b>14</b>
10.1	Creating the Servlet . . . . .	14
10.2	Creating the WAR Archive . . . . .	16
10.3	Adapting the Code for Demo1 . . . . .	17
10.4	Starting the Client . . . . .	17

<b>11 Capturing User Input</b>	<b>17</b>
11.1 Creating the VoiceXML file . . . . .	17
11.2 Creating the Grammar . . . . .	18
11.3 Writing the Client . . . . .	19
11.4 Starting the Client . . . . .	20
<b>12 Builtin Grammars</b>	<b>20</b>
<b>13 Configuration</b>	<b>20</b>
13.1 JNDI Port . . . . .	20
13.2 Talking Java . . . . .	21

### Abstract

This documents describes the API of JVoiceXML from the user's point of view. It provides information about the coding of clients for the JVoiceXML voice browser.

## 1 Introduction

JVoiceXML is a free VoiceXML [8] implementation written in the JAVA programming language with an open architecture for custom extensions. It offers a library for easy VoiceXML document creation and a VoiceXML interpreter to process VoiceXML documents. Demo implementation platforms are supporting JAVA standard APIs such as JSAPI [6] and JTAPI [6].

JVoiceXML is hosted at SourceForge [4] as an open source project. You find everything that is related to this project under <http://sourceforge.net/projects/jvoicexml/>. The work on the browser is still in progress and not all tags are supported, yet. You are invited to help us finishing the work to make this project a success.

This document provides information about the installation and configuration of the JVoiceXML voice browser and how to write VoiceXML applications for this browser. It is assumed that readers are familiar with the concepts of VoiceXML and Java programming.

This document refers to UNIX and Windows systems. JVoiceXML will work with 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 or should be added.

## 2 Copyright

JVoiceXML uses the GNU library general public license [2]. This is mentioned in all our source files as a unique header. 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.

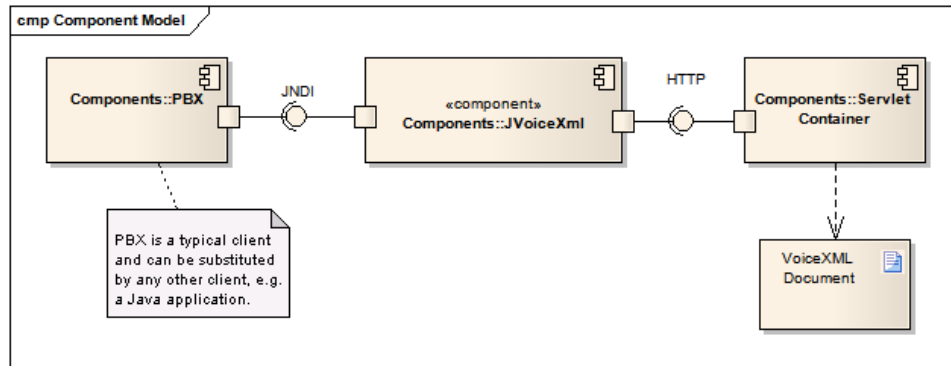


Figure 1: Basic architecture of JVoiceXML

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 Architectural Overview

Before going into detail the general architecture and concepts are presented. The basic architecture is shown in figure 1.

The VoiceXML documents are stored in a web server or a servlet container and are accessed, e.g., via the HTTP protocol. JVoiceXML runs as a standalone server and retrieves the documents from the servlet container.

Clients use the Java Naming and Directory Interface (JNDI) [3] to access JVoiceXML. They can also initiate calls for an application using this technology. Currently there is no telephony support, but users can call applications from their own Java programs. The way this is done is described in the following sections.

Conceptually JNDI allows to connect to a centralized running JVoiceXML server. However this does not make much sense for the current demo implementation, since the speaker and the microphone of the JVoiceXML server is used for speech output and input.

### 4 Required Software

JVoiceXML is written in JAVA and you will at least need a JAVA compiler, an editor or preferably a JAVA IDE, see section 4.1, and ANT, see section 4.3, to run the browser and build the binaries for the clients. Tomcat [1] from the Apache Software Foundation can be used as a servlet container.

## 4.1 IDE

You can use the IDE of your choice to edit the sources and compile the demos. 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 at least J2SE 1.6. The demos use ANT 1.7 for compilation. ANT is not required but used as a means of IDE independent project setup.

## 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>.

## 4.3 ANT

The demos are being built by an ANT build file to keep it IDE independent. 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>.

Nearly all IDEs feature an ANT integration. This allows to use the scripts with your favorite IDE.

## 4.4 Tomcat

VoiceXML is designed to access documents via the HTTP protocol among others. This guide uses Tomcat 5.5 [1] for this purpose. Tomcat can be obtained from <http://tomcat.apache.org>. You can also use the servlet container of your choice.

# 5 Installation

You can download the compiled voice browser as `jvxml-VERSION.zip` from <http://jvoicexml.sourceforge.net/downloads.htm>. `VERSION` has to be replaced by the used version number, e.g. `0.7.0.GA`. Unpack the zipped distribution file and open a command prompt in that directory. Call the installer

```
1 java -jar jvxml-install-VERSION.jar
```

For windows double-clicking the jar should do the trick.

This will install the browser into a directory of your choice. In the rest of this document this directory will be referred as `JVOICEXML_HOME`.

JVoiceXML is shipped with different implementation platforms. Install only those platforms that you intend to use. The configuration issues of each platform is described in the following sections.

It is also possible to install everything and drop those configuration files from the \$JVOICEXML\_HOME/config folder that you do not need. You can simply create a subfolder unused in that directory and move the unused configuration files to this folder. The configuration files follow the naming convention <platform>-implementation.xml. The following section gives a first insight into the overall configuration concept.

## 5.1 Implementation Platform Configuration

With the release of JVoiceXML 0.7.0.GA a new configuration concept was introduced which allows for a more flexible and modular configuration. Each implementation platform can add custom libraries without the need to adapt the startup script. The following code snippet shows the configuration of the text based implementation platform.

```

<?xml version="1.0" encoding="UTF-8" ?>
<implementation
  xmlns:beans="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5  xsi:noNamespaceSchemaLocation=
   "jvxml-implementation-0-7.xsd">
  <classpath>lib/jvxml-text.jar</classpath>
  <classpath>lib/jvxml-client-text.jar</classpath>
  <beans:bean class=
10  "org.jvoicexml.implementation.text.TextPlatformFactory">
    <beans:property name="instances" value="1" />
  </beans:bean>
</implementation>

```

The configuration introduces two new Java archives to the classloader: `ib/jvxml-text.jar` and `ib/jvxml-client-text.jar`.

### 5.1.1 Classloader repositories

The Java archives that are introduced by the configuration are loaded in isolated classloader repositories. Java regards two classes to be different if they are loaded from different classloaders these libraries must share the same classloader. On the one hand this is an advantage since it offers the opportunity to have different versions of the same library active in different classloader repositories, on the other hand this can be a drawback since we want to share the libraries among different configuration settings, e.g. a callmanager configuration and an implementation platform configuration. Sharing the same classloader repository can be achieved by adding the following line to the configuration file

```
<repository>name</repository>
```

*name* should be replaced a proper name for the repository, e.g. *text* for the text based implementation platform.

A closer look at certain configuration issues is given in section 13.

## 5.2 JSAPI 1.0 implementation platform

The JSAPI 1.0 implementation platform targets JSAPI 1.0 compliant speech recognizers and synthesizers. As a first start JVoiceXML is shipped with Sphinx 4 and FreeTTS.

It is also possible to use other speech recognizers and synthesizers. An example how to enable Talking Java for JVoiceXML is described in section 13.2.

## 5.3 JSAPI 2.0 implementation platform

The JSAPI 2.0 implementation platform targets JSAPI 2.0 compliant speech recognizers and synthesizers. As a first start JVoiceXML is shipped with a first approach to use Sphinx 4 and FreeTTS with this new API.

This is an implementation of a draft specification developed under the Java Community Process (JCP) and is made available for testing and evaluation purposes only. The code is not compatible with any specification of the JCP.

Note that you still need to download `jsapi2.jar` from <http://www.conversations.com> and copy it to the `$JVOICEXML_HOME/lib` folder. Otherwise you will get an exception when you start the voice browser.

## 5.4 JTAPI implementation platform

The JTAPI implementation platform can be used in addition to any other implementation platform to enable telephony support. Currently there are some basic tests with the JSAPI 2.0 implementation platform, but this one needs some more programming.

## 5.5 MRCPv2 implementation platform

The MRCPv2 implementation platform targets MRCPv2 compliant speech recognizers and synthesizers. This platform is currently not working. It also hinders JVoiceXML from a proper startup. If you are not interested in debugging of this implementation platform it is recommended to remove this configuration file from the config folder.

## 5.6 Text implementation platform

The text implementation platform can be used to have a string based access to the voice browser.

# 6 Starting the Voice Browser

After the installation, the browser is ready to use. The `bin` folder contains the files to start the browser. The relevant files depend on your operating system and are described in the following sections.

Make sure that you have your configuration right as described in section 5.1 and that you downloaded and installed the missing jars.

## 6.1 Linux

The shell script `startup.sh` located in the `bin` folder of your JVoiceXML installation can be used to start the browser.

It is written to work independent to the current folder. Simply call

```
sh JVOICEXMLHOME/bin/startup.sh
```

After the start lots of debug information will be displayed. It may take a while until the TTS engine and the recognizer are launched. The voice browser can be used, if you see the message

```
VoiceXML interpreter <version> (Build <number>) started.
```

## 6.2 Windows

The windows executable `JVoiceXML.exe` located in the `bin` folder of your JVoiceXML installation can be used to start the browser.

The executable is simply a wrapped Java call and should also work with a double-click in the windows explorer.

From The command line prompt, call

```
JVOICEXMLHOME\bin\JVoiceXML.exe
```

If you start the browser from the windows explorer, a command prompt will open. After the start lots of debug information will be displayed. It may take a while until the TTS engine and the recognizer are launched. The voice browser can be used, if you see the message

```
VoiceXML interpreter <version> (Build <number>) started.
```

## 7 Shutdown of the Voice Browser

The `bin` folder also contains the files to stop the browser. The relevant files depend on your operating system and are described in the following sections.

Please avoid to stop the browser using CTRL-C or by closing the window. If you have JNDI configured JVoiceXML starts the `rmiregistry`. The registry may not shutdown properly if you closed the voice browser this way and may keep the configured port active. This will result in some error messages if you restart JVoiceXML.

### 7.1 Linux

The shell script `shutdown.sh` located in the `bin` folder of your JVoiceXML installation can be used to stop the browser.

It is written to work independent to the current folder. Simply call

```
sh JVOICEXMLHOME/bin/shutdown.sh
```

This will make an RMI call to the voice browser and asks it to shutdown.

### 7.2 Windows

The windows executable `Shutdown.exe` located in the `bin` folder of your JVoiceXML installation can be used to stop the browser.

The executable is simply a wrapped Java call and should also work with a double-click in the windows explorer.

From The command line prompt, call

```
JVOICEXMLHOME\bin\Shutdown.exe
```

This will make an RMI call to the voice browser and asks it to shutdown.

## 8 Running the Demos

The browser comes with some demo programs. You'll find them in the directory `JVOICEXML_HOME/demo`. Use the IDE of your choice and explore there contents. Some features of the browser can become more clear with them.

The demo programs feature an ANT script which can be used for starting. There may be some properties that need to be overwritten in the installed version. For this purpose there is a template `ant.properties` of relevant properties in the folder `JVOICEXML_HOME/demo/config-props`. To override these properties copy the file `ant.properties` to the folder `JVOICEXML_HOME/demo/personal-props`. This way you are able to keep the original settings but also override custom values.

In most cases it should be sufficient to change to each demo directory and call

```
ant run
```

The procedure described above will not work for the *HelloWorldServlet-Demo*. In this case you have to add the location of `javax.servlet-api.jar` to the `voice.xml.properties` by adjusting the property `javax.servlet.lib.dir`.

Before you can run this demo, call

```
ant war
```

to create a war archive that must be deployed to your servlet container before running the demo.

## 9 A first TTS example

This first example shows how VoiceXML documents are accessed from a servlet container or a web server and how clients can start the application.

It is also possible to have the VoiceXML files in the file system. In this case you have to use a URL using the file scheme, e.g. `file:///home/user/text.vxml`. For this guide we follow the W3C specification to retrieve the VoiceXML documents via the HTTP protocol.

### 9.1 Creating the VoiceXML file

This first example is very simple. It just echos a 'hello world'. Create a file `hello.vxml` with the following content:

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml xmlns="http://www.w3.org/2001/vxml" version="2.1">
  <form>
    <block>Hello World!</block>
  </form>
</vxml>
```

Copy this file to a directory of your web server that can be accessed by a browser. For Tomcat create a directory `demo1` in the `$CATALINA_HOME/web-apps` directory and copy the VoiceXML file to this directory. In order to make this an accessible web application, create the empty sub-folder `WEB-INF` in `demo1`.

Now, try to access the file in your browser. For Tomcat this is `http://localhost:8080/demo1/hello.vxml`. If all went well the contents of this file is displayed in the browser. In some cases the file might be offered for download. This is the default behaviour of your browser if it can not determine the type of the file. Download the file and open it in your favourite editor to verify that this is your VoiceXML code.

## 9.2 Writing the Client

A client is a program that remotely calls JVoiceXML and initiates calls. Create a Java file `Demo1.java` with the following content:

```
public class Demo1 {
    public static void main(String [] args) {
    }
}
4
```

First, we need to connect to the JVoiceXML voice browser. JVoiceXML uses JNDI over RMI [5, 7] for this purpose. The following code snippet shows how to obtain a remote reference to the main entry for all client applications `org.jvoicexml.JVoiceXml`:

```
1 import javax.naming.Context;
import javax.naming.InitialContext;

import org.jvoicexml.JVoiceXml;
...
6 public static void main(String [] args) {
    Context context;
    try {
        context = new InitialContext();
    } catch (javax.naming.NamingException ne) {
11     ne.printStackTrace();
        System.exit(-1);
    }

    JVoiceXml jvxml;
16    try {
        jvxml = (JVoiceXml) context.lookup("JVoiceXml");
    } catch (javax.naming.NamingException ne) {
        ne.printStackTrace();
        System.exit(-1);
21    }
    ...
}
```

In line 9, a `Context` is created to access JNDI resources. The settings how to do this are obtained from a file named `jndi.properties` which is in the `CLASSPATH`. `jndi.properties` has the following contents:

```
java.naming.factory.initial=\
    com.sun.jndi.rmi.registry.RegistryContextFactory
3 java.naming.provider.url=rmi://localhost:1099
java.naming.rmi.security.manager=true
```

The location of JVoiceXML is stored in the property `java.naming.provider.url`. If you want to access JVoiceXML on a different computer you have to replace `localhost` with the IP address or name of that computer.

The classes that are required to access JVoiceXML, like `org.jvoicexml.JVoiceXml`, are part of the `jvxml-client.jar`, which can be found in

the `lib` folder of your JVoiceXML installation. This jar contains all classes, that you need to write client applications.

If you are using Java 5 you will have to add the Java API for XML streaming. This became part of Java 6 as JSR 173. These are `jsr173_1.0.jar` and `sjsxp.jar`. You can find a copy of these libraries in the `lib` folder of your JVoiceXML installation.

Next, we call the browser to process the application. This is done by creating a `org.jvoicexml.Session` object.

```

1  ...
import org.jvoicexml.Session;
...

6  public static void main(String[] args) {
    ...
    JVoiceXml jvxml;
    try {
        jvxml = (JVoiceXml) context.lookup("JVoiceXml");
    } catch (javax.naming.NamingException ne) {
11     ne.printStackTrace();
        System.exit(-1);
    }

    final Session session =
16     jvxml.createSession(null);

    final URI uri;
    try {
        uri =
21     new URI("http://localhost:8080/demo1/hello.vxml");
    } catch (URISyntaxException e) {
        e.printStackTrace();

        System.exit(-1);
26     }

    try {
        session.call(uri);

31     session.waitForSessionEnd();

        session.close();
    } catch (org.jvoicexml.event.JVoiceXMLEvent e) {
        e.printStackTrace();

36     System.exit(-1);
    }
    ...

```

The argument on the `createSession` method must be `null`. It is reserved for future use, once we have telephony support. The argument for `call` must point to the URI of the root document of your application.

### 9.3 Starting the Client

The JNDI implementation of JVoiceXML is based on RMI, and the implementation for the used interfaces are obtained by RMI dynamic code download. This means that you have to provide the location of the library with the implementation of the interfaces and a security policy file.

For the start this security policy file `jvoicexml.policy` allows everything to the remote user:

```
1 grant {
    permission java.security.AllPermission;
};
```

A more restrictive policy can be

```
2 grant {
    permission java.util.PropertyPermission
    "jvoicexml.vxml.version", "read";
    permission java.util.PropertyPermission
    "jvoicexml.xml.encoding", "read";
    7 permission java.net.SocketPermission
    "127.0.0.1:1024-", "connect, resolve";
    permission java.io.FilePermission
    "${JVOICEXMLHOME}/lib/-", "read";
};
```

The location of the policy is provided by the following environment property

```
-Djava.security.policy=jvoicexml.policy \
```

Once you start `Demo1` it connects to JVoiceXML and starts processing the application. If you are successful you should hear a synthesized voice speaking *Hello World*. The application terminates when the processing finishes.

## 10 Creating VoiceXML using the Tag Library

JVoiceXML features a strong tag library to author VoiceXML documents. In this section we will write a small servlet returning the VoiceXML document that was used in section 9 using this library.

### 10.1 Creating the Servlet

Our basic skeleton for a servlet looks as follows:

```

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
4 import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class HelloServlet extends HttpServlet {
    public void doGet(HttpServletRequest request,
9         HttpServletResponse response)
        throws ServletException, IOException {
    }
}

```

Our code goes into the `doGet()` method of the servlet. The tag library is located in the package `org.jvoicexml.xml`. Hence you have to add `jvxml-xml.jar` to the `CLASSPATH`.

Before using the classes you have to import the required classes by adding

```
import org.jvoicexml.xml.*;
```

Then, the VoiceXML document can be created by adding

```

    public void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
4         Vxml vxml = document.getVxml();
        Form form = vxml.appendChild(Form.class);

        Block block = form.appendChild(Block.class);
        block.addText("Hello World!");
9     }

```

Each tag has a corresponding class in the tag library and has also convenient methods to set and get the allowed attributes.

A child tag can be added using the scheme

```
1 ChildTag child = parentTag.appendChild(ChildTag.class);
```

`ParentTag` and `ChildTag` have to be replaced by the concrete class. If a child tag is not allowed for a parent tag, a `IllegalArgumentException` is thrown.

Next we are going to send the created document to the servlet response stream. This is done by adding the following code right after the document code:

```

        response.setContentType("text/xml");
        final String xml = document.toString();
        final PrintWriter out = response.getWriter();
4        out.println(xml);

```

A string representation of the created document can be obtained via the `toString()` method. `JVoiceXML` uses the Java API for XML streaming for

this purpose which is part of Java 6. If you are using Java 5, you will have to add `jsr173_1.0_api.jar` and `sjsxp.jar`.

## 10.2 Creating the WAR Archive

Servlets are distributed as a war archive. The description for the servlet container, e.g. Tomcat, is located in the `web.xml` file. This file has the following content for our example:

```

1 <?xml version="1.0" encoding="ISO-8859-1"?>
  <!DOCTYPE web-app
    PUBLIC
    "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
6    "http://java.sun.com/dtd/web-app_2_3.dtd">
  <web-app>
    <display-name>JVoiceXML HelloWorld Demo</display-name>
    <description>
11      Demo for servlet based VoiceXML creation.
    </description>
    <servlet>
      <servlet-name>JVoiceXMLHelloWorldDemo</servlet-name>
16      <servlet-class>
        HelloServlet
      </servlet-class>
    </servlet>
    <servlet-mapping>
21      <servlet-name>JVoiceXMLHelloWorldDemo</servlet-name>
      <url-pattern>/helloworld</url-pattern>
    </servlet-mapping>
  </web-app>

```

The file is stored in the WAR archive `hello.war`. This archive has the following structure

```

+ web.xml
+ WEB-INF
  + classes
    + HelloServlet.class
5  + lib
    + jvxml-xml.jar
    + jsr_173_1.0_api.jar
    + sjsxp.jar

```

Copy the created war archive to the `$CATALINA_HOME/webapps` directory and restart Tomcat.

### 10.3 Adapting the Code for Demo1

Demo1 from section 9 has to be adapted to point to the URL of our servlet. Change the line

```

uri =
2   new URI("http://localhost:8080/demo1/hello.vxml");

```

to

```

uri =
   new URI("http://localhost:8080/hello/helloworld");

```

### 10.4 Starting the Client

To start the adapted client follow the steps as they are described in section 9.3.

## 11 Capturing User Input

This example shows how JVoiceXML can be used to capture user input.

### 11.1 Creating the VoiceXML file

We use the same environment as introduced in section 9.1. Here we use the source folder `demo2`. The demo asks the user a question with the possible answers *Yes* and *No*.

Our VoiceXML code in the file `input.vxml` looks like this:

```

<?xml version="1.0" encoding="UTF-8" ?>
<vxml xmlns="http://www.w3.org/2001/vxml" version="2.1"
3   xml:base="http://localhost:8080/demo2/">
  <form>
    <field name="answer">
      <grammar src="yesno.gram" type="application/x-jsgf"/>
      <block>Do you like this example?</block>
8     <noinput>
        Please say something.
        <reprompt/>
      </noinput>
      <nomatch>
13     Please say yes or no.
        <reprompt/>
      </nomatch>
      <filled>
18     <if cond="answer=='yes'">
        You like this example.
      <else />
        You do not like this example.

```

```

23     </if>
        </filled>
    </field>
</form>
</vxml>

```

## 11.2 Creating the Grammar

The VoiceXML code above relates to the grammar `yesno.gram` in JSGF format. It is also possible to have the grammar in SRGS XML format.

Create a file `yesno.gram` with the following content:

```

grammar yesno ;
public <yesno> = yes | no

```

Add the grammar file to your war archive.

JVoiceXML is shipped with sphinx4 as a demo speech recognizer. Unfortunately, the JSAPI interfaces seems to be buggy and it is not possible to dynamically load or activate grammars. All grammars that are defined at start are active throughout the application. JVoiceXML does not have any limitations in that way. You will be able to use grammar activation as it is meant to be, if you use your own JSAPI compliant recognizer.

Now, edit the file `sphinx4.config.xml` and tell sphinx about your new grammar.

Look for the following code snippet

```

2 <component name="jsgfGrammar"
    type="edu.cmu.sphinx.jsapi.JSGFGrammar">
    <property name="dictionary" value="dictionary"/>
    <property name="grammarLocation"
        value="file:config/" />
    <property name="grammarName" value="movies" />
7 <property name="logMath" value="logMath" />
</component>

```

and replace `movies` by `yesno`.

Place a copy of the grammar file in the `$JVOICEXML_HOME/config` folder so that sphinx4 is able to find it.

```

2 <component name="jsgfGrammar"
    type="edu.cmu.sphinx.jsapi.JSGFGrammar">
    <property name="dictionary" value="dictionary"/>
    <property name="grammarLocation"
        value="file:config/" />
    <property name="grammarName" value="yesno" />
7 <property name="logMath" value="logMath" />
</component>

```

Afterwards you will have to restart JVoiceXML.

### 11.3 Writing the Client

The client for this demo looks pretty much like the client for the *Hello World!* example.

Copy the file `Demo1.java` into a file `Demo2.java` and adapt the URI to point to the `demo2`.

```
...
2 import org.jvoicexml.Session;
...

    public static void main(String [] args) {
    ...
7     JVoiceXml jvxml;
    try {
        jvxml = (JVoiceXml) context.lookup("JVoiceXml");
    } catch (javax.naming.NamingException ne) {
12        ne.printStackTrace();
        System.exit(-1);
    }

    final Session session =
17        jvxml.createSession(null);

    final URI uri;
    try {
        uri =
22        new URI("http://localhost:8080/demo2/input.vxml");
    } catch (URISyntaxException e) {
        e.printStackTrace();

        System.exit(-1);
    }

27    try {
        session.call(uri);

        session.waitForSessionEnd();

32        session.close();
    } catch (org.jvoicexml.event.JVoiceXMLEvent e) {
        e.printStackTrace();

37        System.exit(-1);
    }
...

```

## 11.4 Starting the Client

Start the Demo2 application as you did for Demo1, refer to section 9.3.

You will be prompted *Do you like this example?*. Now you can answer either *yes* or *no*. Depending on what you say you will hear the corresponding statement.

Please use a headset when trying this example. Since the voice browser uses the speaker and the microphone of you PC it may happen that the output of the synthesizer is being recognized as your input by mistake.

## 12 Builtin Grammars

In section 11.2 we manually created the grammar to define the valid user input. Platforms can support fundamental grammars, the so-called *builtin* grammars. Currently JVoiceXML provides initial support for two of them:

- boolean
- digit

The parameters follow the specification of [8] appendix P. The URL must be of the following form:

```
1 builtin://<mode>/<type>[? parameters ]
```

where *mode* is one of *dtmf* or *voice* and *type* denotes one of the types mentioned above.

An grammar using a boolean type with 7 as the value for *yes* and 9 meaning *no* would look as follows:

```
<grammar src="builtin:dtmf/boolean?y=7;n=9"/>
```

Currently JVoiceXML is not able to evaluate the tags within a grammar, so you will have to check for 7 and 9 in your conditions for the moment.

## 13 Configuration

After the installation, JVoiceXML should run out of the box. However, there may be some circumstances, where it is necessary, to adapt the configuration.

### 13.1 JNDI Port

The remote access for clients is based on RMI, using the default RMI port. This can conflict with other applications that also use this technology, like JBoss.

If you want to change the RMI port for JVoiceXML, you have to make changes in two configuration files that you can find in the folder `$JVOICEXML_HOME/config`.

In the file `jvxml-jndi.xml` you have to adapt the `port` attribute in following section

```

4 <?xml version="1.0" encoding="UTF-8"?>
  <jndi xmlns:beans="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="jvxml-jndi-0-7.xsd">
    <repository>text</repository>
    <classpath>lib/jvxml-jndi.jar</classpath>
    <beans:bean id="org.jvoicexml.JndiSupport"
      class="org.jvoicexml.jndi.JVoiceXmlJndiSupport">
9     <beans:property name="registry">
      <beans:bean id="registry"
        class="org.jvoicexml.jndi.JVoiceXmlRegistry">
14     <beans:property name="port" value="1099" />
      </beans:bean>
    </beans:property>
    </beans:bean>
  </jndi>

```

In addition you have to adapt the file `jndi.properties`. Change the port to the same value as above.

```
java.naming.provider.url=rmi://localhost:1099
```

Do not forget to do the same in the `jndi.properties` file of your clients.

## 13.2 Talking Java

The quality of the Sphinx and FreeTTS is not very high. If you are using windows you might want to install TalkingJava from CloudGarden <http://www.cloudgarden.com>. This is a cheap commercial JSAPI 1.0 wrapper for the Microsoft Speech Engine.

In order to use it, you need to have the JSAPI 1.0 implementation platform of JVoiceXML installed.

After the installation of TalkingJava you will have to do the following changes to use the Microsoft Speech Engine.

Copy the `cgjsapi163.dll` to the `$JVOICEXML_HOME/bin` folder and copy the `cgjsapi.jar` to the `$JVOICEXML_HOME/lib` folder.

Make a copy of the file `jsapi10-implementation.xml` in the directory `$JVOICEXML_HOME/config` to another folder and replace the content with the following:

```

5 <?xml version="1.0" encoding="UTF-8"?>
  <implementation
    xmlns:beans="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="
      "jvxml-implementation-0-7.xsd">
    <classpath>lib/jvxml-jsapi1.0.jar</classpath>
    <classpath>lib/cgjsapi.jar</classpath>

```

```
10 <beans:bean class=
    "org.jvoicexml.implementation.jsapi10.Jsapi10SynthesizedOutputFactory">
    <beans:property name="instances" value="1" />
    <beans:property name="type" value="jsapi10" />
    <beans:property name="synthesizerModeDescriptorFactory">
    <beans:bean class=
15 "org.jvoicexml.implementation.jsapi10.JVoiceXmlSynthesizerModeDescFactory">
        </beans:bean>
    </beans:property>
</beans:bean>

20 <beans:bean class=
    "org.jvoicexml.implementation.jsapi10.Jsapi10AudioFileOutputFactory">
    <beans:property name="instances" value="1" />
</beans:bean>

25 <beans:bean class=
    "org.jvoicexml.implementation.jsapi10.Jsapi10SpokenInputFactory">
    <beans:property name="type" value="jsapi10" />
    <beans:property name="instances" value="1" />
    <beans:property name="recognizerModeDescriptorFactory">
30 <beans:bean class=
        "org.jvoicexml.implementation.jsapi10.JVoiceXmlRecognizerModeDescFactory">
        </beans:bean>
    </beans:property>
</beans:bean>

35 <beans:bean class=
    "org.jvoicexml.implementation.jsapi10.Jsapi10TelephonyFactory">
    <beans:property name="instances" value="1" />
</beans:bean>
40 </implementation>
```

## Document history

Version	Comment	Author	Date
0.1	Initial Release	Dirk Schnelle	04/24/2006
0.2	First demo	Dirk Schnelle	04/26/2006
0.3	Architectural overview	Dirk Schnelle	04/27/2006
0.4	Running the demos	Dirk Schnelle	07/20/2006
0.4.1	Adaption to refactoring of 0.5.5	Dirk Schnelle	03/07/2007
0.5	Started user input example	Dirk Schnelle	03/13/2007
0.6	Adaption to 0.6, added VoiceXML creation demo	Dirk Schnelle	06/05/2008
0.7	Adaption to 0.7.0.GA, added TalkingJava configuration	Dirk Schnelle-Walka	06/18/2009
0.7.1	Adaption to 0.7.1.GA, added description for builtin grammars	Dirk Schnelle-Walka	08/04/2009
0.7.1.1	Added description for platform configuration	Dirk Schnelle-Walka	08/05/2009

## References

- [1] Apache Tomcat. <http://tomcat.apache.org>.
- [2] GNU. GNU library general public license. <http://www.opensource.org/licenses/lgpl-license.php>.
- [3] Sun. <http://java.sun.com/products/jndi/>.
- [4] SourceForge.net. <http://sourceforge.net>.
- [5] SUN. Java Remote Method Invocation (Java RMI). <http://java.sun.com/products/jdk/rmi/>.
- [6] SUN. Java Speech API 1.0 (JSAPI). <http://java.sun.com/products/java-media/speech/forDevelopers/jsapi-doc/index.html>.
- [7] SUN. RMI Registry Service Provider for the Java Naming and Directory Interface (JNDI). <http://java.sun.com/j2se/1.5.0/docs/guide/jndi/jndi-rmi.html>.
- [8] W3C. Voice Extensible Markup Language (VoiceXML) Version 2.0. <http://www.w3.org/TR/voicexml20/>, March 2004.