|  |
| --- |
| [Sign In/My Account](http://onjava.com/cs/user/login?x-redirect=/cs/user/home) | [View Cart](http://epoch.oreilly.com/shop/cart.orm) |

* [O'Reilly Home](http://oreilly.com/)
* [Community](http://community.oreilly.com/)
* [Books & Videos](http://oreilly.com/store/)
* [Safari Books Online](http://safari.oreilly.com/?cid=orm-nav-global)
* [Conferences](http://conferences.oreillynet.com/)
* [Training](http://training.oreilly.com/)
* [School of Technology](http://www.oreillylearning.com)
* [About](http://oreilly.com/about/)

Top of Form

Search 

[Search Tips](http://www.oreillynet.com/search/tips.csp)

Bottom of Form

[Print](http://onjava.com/lpt/a/4062) [Subscribe to ONJava](http://www.oreillynet.com/pub/feed/7) [Subscribe to Newsletters](http://www.oreillynet.com/cs/nl/home)

[ShareThis](javascript:void(0))

**The Java Speech API, Part 1**

by [Mandar S. Chitnis](http://onjava.com/pub/au/1262) and [Lakshmi Ananthamurthy](http://onjava.com/pub/au/1263)  
08/06/2003

The idea of machines that speak and understand human speech has long been a fascination of application users and application builders. With advances in speech technology, this concept has now become a reality. Research projects have evolved and refined speech technology, making it feasible to develop applications that use speech technology to enhance the user's experience. There are two main speech technology concepts -- **speech synthesis** and **speech recognition**.

Speech synthesis is the process of generating human speech from written text for a specific language. Speech recognition is the process of converting human speech to words/commands. This converted text can be used or interpreted in different ways.

Over the course of two articles, we will explore the use of the Java Speech API to write applications that have speech synthesis and speech recognition capabilities. In addition, we will look at the application areas where we can effectively use speech technology.

**Speech Technology Support in Java**

A speech-enabled application does not directly interact with the audio hardware of the machine on which it runs. Instead, there is a common application, termed the Speech Engine, which provides speech capability and mediates between the audio hardware and the speech-enabled application, as shown in Figure 1.

*Figure 1. Speech engine*

Speech engines implemented by each vendor expose speech capabilities in a vendor-specific way. To enable speech applications to use speech functionality, vendors design speech engines that expose services that can be accessed via a commonly defined and agreed-upon Application Program Interface.

**Java Speech API**

This is where the Java Speech API (JSAPI) steps into the picture. The Java Speech API brings to the table all of the platform- and vendor-independent features commonly associated with any Java API. The Java Speech API enables speech applications to interact with speech engines in a common, standardized, and implementation-independent manner. Speech engines from different vendors can be accessed using the Java Speech API, as long as they are JSAPI-compliant.

With JSAPI, speech applications can use speech engine functionality such as selecting a specific language or a voice, as well as any required audio resources. JSAPI provides an API for both speech synthesis and speech recognition.

*Figure 2. The Java Speech API stack*

Figure 2 shows the Java Speech API stack. At the bottom of the stack, the speech engine interacts with the audio hardware. On top of it sits the Java Speech API that provides a standard and consistent way to access the speech synthesis and speech recognition functionality provided by the speech engine. Java applications that need to incorporate speech functionality use the Java Speech API to access the speech engine.

Several speech engines, both commercial and open source, are JSAPI-compliant. Among open source engines, the Festival speech synthesis system is one of the popular speech synthesis engines that expose services using JSAPI. Many commercial speech engines that support JSAPI exist. You can find a comprehensive list of these on the [Java Speech API](http://java.sun.com/products/java-media/speech/) web site.

**Java Speech API: Important Classes and Interfaces**

The different classes and interfaces that form the JSAPI are grouped into three packages:

* javax.speech contains classes and interfaces for a generic speech engine.
* javax.speech.synthesis contains classes and interfaces for speech synthesis.
* javax.speech.recognition contains classes and interfaces for speech recognition.

Before we proceed with writing an application that uses JSAPI, let's explore a few important classes and interfaces in each of these packages.

*Figure 3. JSAPI speech engine interfaces and classes*

**Central**

The Central class is like a factory class that all JSAPI applications use. It provides static methods to enable the access of speech synthesis and speech recognition engines.

**Engine**

The Engine interface encapsulates the generic operations that a JSAPI-compliant speech engine should provide for speech applications. Primarily, speech applications can use methods to perform actions such as retrieving the properties and state of the speech engine and allocating and deallocating resources for a speech engine. In addition, the Engine interface exposes mechanisms to pause and resume the audio stream generated or processed by the speech engine. The Engine interface is subclassed by the Synthesizer and Recognizer interfaces, which define additional speech synthesis and speech recognition functionality.

The JSAPI has been modeled on the event-handling model of AWT components. Hence, events generated by the speech engine can be identified and handled as required. There are two ways to handle speech engine events: through the EngineListener interface or the EngineAdapter class.

Next, let's examine some of the important classes and interfaces of the javax.speech.synthesis package. These will be used quite frequently in speech applications.

*Figure 4. JSAPI speech synthesis interfaces and classes*

**Synthesizer**

The Synthesizer interface encapsulates the operations that a JSAPI-compliant speech synthesis engine should provide for speech applications. Primarily, speech applications can perform actions such as producing speech output (given text input) or stopping speech-synthesis processing. Other related operations are inherited from the Engine interface. The Synthesizer interface provides different sources of text input, ranging from a plain String, to a URL, to a special-purpose markup language called Java Speech Markup Language (JSML, discussed in the next article).

**SynthesizerProperties**

The operations in the SynthesizerProperties interface are used to define runtime properties for the Synthesizer object, including the voice, volume, and pitch for speech synthesis by the Synthesizer object.

**Voice**

The Voice class represents the voice that the Synthesizer object uses to play the speech output. The Voice class also provides methods to obtain metadata information for the voice used for speech synthesis by the Synthesizer object. This metadata includes the name, age, and gender of the voice being used.

Similar to the Engine interface, events generated during speech synthesis can be identified and handled by either implementing the methods in the SpeakableListener interface or using the SpeakableAdapter class.

We will explore the classes and interfaces of the javax.speech.recognition package in the next article.

**"Can you hear me now?" Asks the Duke**

In order to understand the JSAPI better, let's write a simple application that uses the JSAPI to provide speech synthesis capability. We will build a simple text editor using the Java Swing API set and add speech capability to the editor using JSAPI to enable the application to speak the contents of a file. What speech synthesis capabilities do we add to the speech-enabled text editor? As we saw from the previous section, the speech synthesis engine provides different features, such as producing speech output from text, pausing or resuming the speech output, or ending the speech output generation. We can add the following capabilities to the VoicePad editor:

* **Play**: speak the contents of the text editor.
* **Pause**: pause the playing of the speech output.
* **Resume**: resume the playing of the speech output from the last pause.
* **Cancel**: stop the speech output.

The user can invoke any of these speech capabilities by clicking on the relevant menu items (“Play”, “Pause”, “Resume,” or “Cancel”) from the Speech Menu.

To build the speech-enabled text editor, first we will define the user interface (UI) elements that we will need. We can use the text area element as the text editor for our application. For navigation and user interaction, we will define a menu bar with menus and menu options. Since our application functionality consists of two parts – text editor and speech – we will define two sets of menus for the VoicePad application:

* A file menu that supports file operations: creating a new file, opening an existing file, saving the contents of an edited file, and closing a file.
* A speech menu that supports speech synthesis operations: speaking the contents of the text editor, pausing and resuming the speech synthesis operations, and canceling a speech operation in progress.

*Figure 5. Class diagram of the VoicePad application*

Now that we know what is required, let's put together the Java Swing pieces for the application. The primary class of the application is the VoicePad class that extends from the JFrame class. As shown in the class diagram in Figure 5, the VoicePad class will contain all of the methods required for both text editing and speech functionality.

The constructor for the VoicePad application is responsible for initializing the application elements. The constructor invokes the init() method, which performs the initialization of the user interface elements and the speech engine. The JTextArea UI element will be the text editor for our application.

// constructor

public VoicePad()

{

super("Novusware - VoicePad");

setSize(800, 600);

// initialize the application settings

init();

// set up the file selection

fileChooser = new JFileChooser();

fileChooser.setCurrentDirectory(new File("."));

WindowListener wndCloser = new WindowAdapter()

{

public void windowClosing(WindowEvent e)

{

closeSpeechSynthesisEngine();

System.exit(0);

}

};

addWindowListener(wndCloser);

}

// initialization

private void init()

{

textArea = new JTextArea();

ps = new JScrollPane(textArea);

this.getContentPane().add(ps, BorderLayout.CENTER);

textArea.append("Voice-enabled editor.");

initSpeechSynthesisEngine();

this.setJMenuBar(getVoicePadMenuBar());

System.out.println("NOVUSWARE : VoicePad application initialized.");

}

|  |  |
| --- | --- |
| Pages: **1**, [2](http://onjava.com/pub/a/onjava/2003/08/06/jsapi.html?page=2), [3](http://onjava.com/pub/a/onjava/2003/08/06/jsapi.html?page=3) | [Next Page](http://onjava.com/pub/a/onjava/2003/08/06/jsapi.html?page=2) |

**Tagged Articles**

[Be the first to post this article to del.icio.us](http://del.icio.us/post)

**Sponsored Resources**

* [Inside Lightroom](http://digitalmedia.oreilly.com/lightroom/)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Top of Form   |  |  | | --- | --- | | Sign up today to receive special discounts, product alerts, and news from O'Reilly. |  |   Bottom of Form | [Privacy Policy >](http://oreilly.com/oreilly/privacy.html) [View Sample Newsletter >](http://oreilly.com/pdf/sample-newsletter.pdf) | * [View All RSS Feeds >](http://oreilly.com/feeds/) |

|  |  |  |  |
| --- | --- | --- | --- |
| 800-889-8969 or 707-827-7019  Monday-Friday 7:30am-5pm PT ©2011, O'Reilly Media, Inc. All trademarks and registered trademarks appearing on oreilly.com are the property of their respective owners. | * [**About O'Reilly**](http://oreilly.com/about/) * [Academic Solutions](http://oreilly.com/academic/) * [Contacts](http://oreilly.com/contact.html) * [Customer Service](http://oreilly.com/oreilly/cs/) * [Careers](http://oreilly.com/jobs/) * [Press Room](http://press.oreilly.com/) * [Privacy Policy](http://oreilly.com/oreilly/privacy.csp) * [Terms of Service](http://oreilly.com/terms/) * [Writing for O'Reilly](http://oreilly.com/oreilly/author/intro.csp) | * [**Community**](http://oreilly.com/community/) * [Authors](http://oreilly.com/authors/) * [Forums](http://forums.oreilly.com/) * [Membership](https://members.oreilly.com/) * [Newsletters](http://elists.oreilly.com/) * [RSS Feeds](http://oreilly.com/feeds/) * [User Groups](http://ug.oreilly.com/) | * **More O'Reilly Sites** * [igniteshow.com](http://igniteshow.com/) * [makerfaire.com](http://makerfaire.com/) * [makezine.com](http://makezine.com/) * [craftzine.com](http://craftzine.com) * [labs.oreilly.com](http://labs.oreilly.com) * **Partner Sites** * [PayPal Developer Zone](https://www.x.com/community/ppx/devzone) * [O'Reilly Insights on Forbes.com](http://blogs.forbes.com/oreillymedia/) |