Hindawi Publishing Corporation
Advances in Software Engineering
Volume 2012, Article ID 981783, 9 pages
doi:10.1155/2012/981783

Research Article

A Simple Application Program Interface for Saving Java Program Data on a Wiki

Takashi Yamanoue, Kentaro Oda, and Koichi Shimozono

Computing and Communications Center, Kagoshima University, Korimoto, Kagoshima 890-0065, Japan

Correspondence should be addressed to Takashi Yamanoue, yamanoue@cc.kagoshima-u.ac.jp

Received 14 November 2011; Revised 18 January 2012; Accepted 23 January 2012

Academic Editor: Andreas Menychtas

A simple application program interface (API) for Java programs running on a wiki is implemented experimentally. A Java program with the API can be running on a wiki, and the Java program can save its data on the wiki. The Java program consists of PukiWiki, which is a popular wiki in Japan, and a plug-in, which starts up Java programs and classes of Java. A Java applet with default access privilege cannot save its data at a local host. We have constructed an API of applets for easy and unified data input and output at a remote host. We also combined the proposed API and the wiki system by introducing a wiki tag for starting Java applets. It is easy to introduce new types of applications using the proposed API. We have embedded programs such as a simple text editor, a simple music editor, a simple drawing program, and programming environments in a PukiWiki system using this API.

1. Introduction

The Web is currently one of the most important infrastructures. However, managing a web site is not easy, and it is necessary to upload a file to a web server each time a new web page is created or an existing web page is modified. For example, teachers, including university instructors, often use web sites in class. However, preparing web-based learning materials is troublesome. In order to facilitate this task, we use a content management system (CMS), such as wiki software.

A wiki [1] is a web site that allows the easy creation and editing of any number of interlinked web pages via a web browser and can be used as a means of effective collaboration and information sharing. Wikipedia [2] is a well-known wiki site.

The Internet provides a large number of Java applets, through which multimedia data can be used. Such multimedia data can be used on wiki sites. As such, a number of wikis have extensions or plug-ins for embedding Java applets. Saving such Java applet data on a wiki site is advantageous, which makes the wiki more flexible and extensible. This allows more effective collaboration between users.

PukiWiki [3] software is commonly used in Japan. We have constructed an API for applets in order to allow easy and unified data input and output at a remote host. Moreover, we have combined the API and the PukiWiki system by introducing a wiki tag for starting Java applets. The proposed API, which can be used to make the wiki more flexible and extensible, is referred to as the *PukiWiki-Java Connector*.

The PukiWiki-Java Connector enables a number of Java programs to be easily embedded in PukiWiki. We have embedded programs such as a simple text editor, a simple music editor, a simple drawing program, a programming environment, and a voice recorder in PukiWiki. One to three days was required for embedding.

2. Pukiwiki-JAVA Connector

2.1. Outline of Usage. A wiki [1] is a web site whose users can add, modify, or delete its content via a web browser using a simplified markup language or a rich-text editor. Wikis are often used for collaboration. Users of a wiki create and edit each others content when they collaborate using the wiki. Users of a wiki often would like to use not only texts but also other rich media content such like figures or sounds for effective collaboration. Many of wikis can have such contents. However, it was hard to add and modify such contents via

web browsers. PukiWiki-Java connector is a tool to solve such problem.

To implement our proposed connector to a wiki, we need a wiki platform, which is simple to extend and easy to deploy. We choose PukiWiki [4] as the platform. PukiWiki is simple to deploy than many other wikis because it needs minimized requirements; it even requires no data base engines such as MySQL. Installing Pukiwiki is just extracting PukiWiki tarball to the root directory of the web page. PukiWki also puts minimized requirements for its plugin specification [4]; it defines a relatively small set of rules to create plugins. So There are a lot of plug-ins for PukiWiki. PukiWiki supports not only Japanese language but also English, Chinese, and French.

The PukiWiki-Java Connector enables a Java program to input and output data on a remote host. A PukiWiki system is run at the host, and the system starts the Java program as a PukiWiki plug-in.

Figure 1 shows a use case diagram of the PukWiki-Java Connector and related items. A programmer creates the Java program for PukiWiki using the PukiWiki-Java Connector in a programming environment such as Eclipse. A web administrator uploads the Java program to a directory in the PukiWiki system. A user writes a PukiWiki web page that includes a Java program plug-in using a web browser on the user's side. A user uses the Java program by displaying the web page in a web browser. The user can save program data in the page. The data is loaded to the Java program when the user, or another user, next uses the program. Users can collaborate through Java programs using the PukiWiki-Java Connector.

It is easy to copy the saved data from the page to another page of the PukiWiki system that has the same Java program by coping and pasting the page.

Occasionally, we would like to use a wiki for a closed group. Basic authentication is an easy way to realize this task. The PukiWiki-Java Connector can also be used for a web site with basic authentication. When a Java program with the PukiWiki-Java Connector saves data to or loads data from a web page with basic authentication [5], a dialog box pops up, prompting the user to input an ID and corresponding password. Figure 2 shows an example of the dialog box.

Figure 3 shows an outline of the use of the PukiWiki-Java Connector. In this figure, Java program X is a program that uses PukiWiki-Java Connector. In addition, "#jcon(X)" is the plug-in for starting Java program X. Java program X is downloaded to a web browser at the user side when the wiki page with "#jcon(X)" is displayed by the web browser. The data of Java program X can be saved to or loaded from a page.

In order to save data to or load data from a web page, the PukiWiki-Java Connector provides an interface with the following methods. Java program X can implement the interface for saving and loading data:

```
public String getOutput();
public void setInput(String x);
public void setSaveButtonDebugFrame
(SaveButtonDebugFrame f);
```

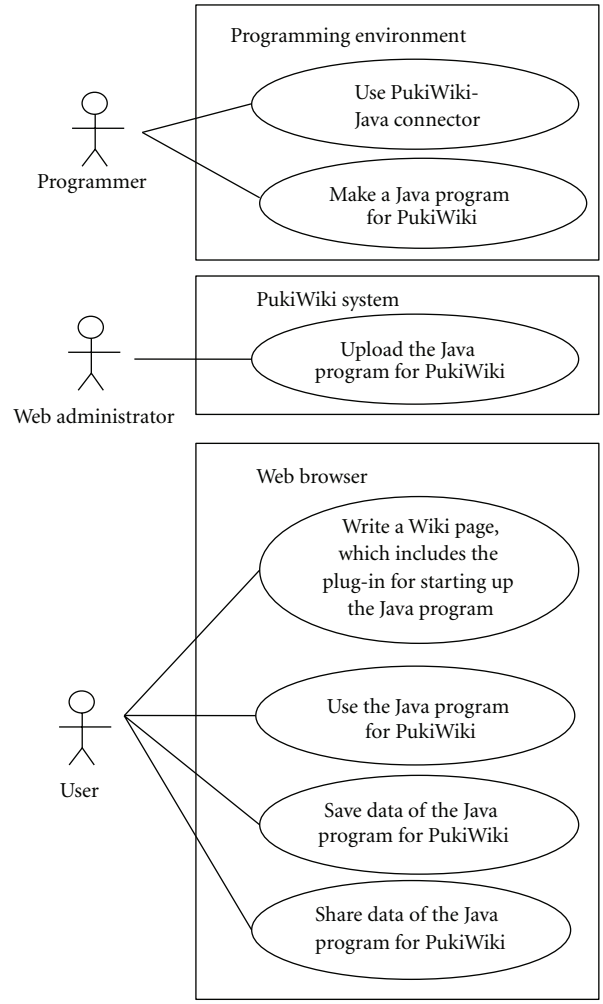

Figure 1: Use case diagram of the PukiWiki-Java Connector.

Here, get Output() is called from the PukiWiki-Java Connector when the data of Java program X is saved. The programmer of Java program X writes the code for returning the data to be saved as String type data in this override method. Moreover, setInput(String x) is called from the PukiWiki-Java Connector when data on the wiki page is loaded to this program. The programmer writes the code for inputting the data in this override method. Then, setSaveButtonDebugFrame is used for debugging. *Factory method pattern* [6] is used to facilitate the embedding of several Java programs to PukiWiki. Figure 4 shows a class diagram of factory method pattern for the PukiWiki-Java Connector. In this figure, PukiWikiApplet is the abstract class that is started from the PukiWiki system. PukiWikiJavaApplication is the interface for a Java program that uses the PukiWiki-Java Connector. MyApplet is a concrete class for creating a Java program, and X is a concrete class of Java program X. MyApplet.class and X.class should be in the directory of *./javaApplications/bin/application/X/*.

2.2. Outline of Implementation. We assume that *wiki page X* contains the wiki tag, which starts the *Java program X* with

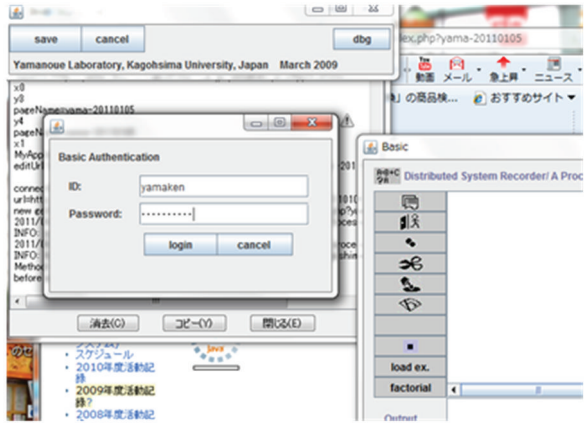

Figure 2: Basic authentication dialog box.

Pukiwiki-Java connector. *Data of X* is loaded from the page and it is saved to the page. Wiki page X has its *page name*.

Starting a Java Program. In order to start the Java program with the PukiWiki-Java Connector, the PukiWiki system must have the PHP code for the #jcon(X) plug-in. Algorithm 1 shows the PHP code. This code starts the applet at the path *./javaApplication/bin/applications/X/MyApplet*. Then, adding or modifying a PukiWiki system plug-in is not required for embedding a new Java application using the PukiWiki-Java Connector.

When the Java program X is started from the wiki page X, the URL of the page and the page name are acquired by the applet. The URL is used for loading data from the page. The URL and the page name are used for saving data to the page.

Loading Data. The source text of the wiki page X is acquired by *SaveButtonDegugFrame* class after the starting using the URL. The class looks for the saved data by parsing the page. The data is located in the page after the line of the wiki tag "#jcon(X)", and the data is enclosed by the tags "<pre>" and "</pre>". The tags show preformatted text in HTML. If the data was found out, the data is extracted from the wiki page, and it is sent to the Java program X using the set Input(x) method of the program. The *GetMethod* class of *Apache HttpClient* [7] is used for acquiring the page.

Saving Data. When the "save" button of the PukiWiki-Java connector is clicked, the source text of the wiki page X in the editing mode is acquired by the *SaveButtonDebugFrame* class using the URL for the editing mode and the *GetMethod* class. In the case of PukiWiki, the URL of a wiki page in the editing mode is represented by Algorithm 1.

The source text of the page for editing mode contains a pair of form tags <form ... > and </form>. The pair text area tags <textarea ... > and </textarea> is enclosed by the form tags. The text between the pair of text area tags shows the current source text of the wiki page in PukiWiki page syntax. So the text between the pair of text area tags has the

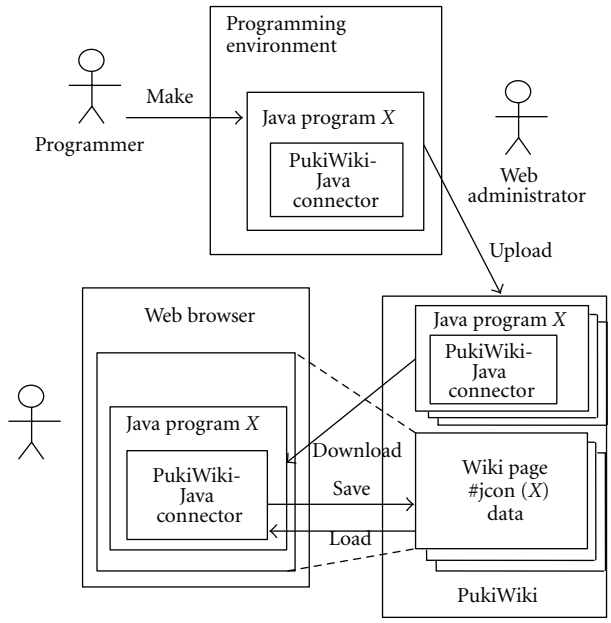

Figure 3: Outline of the use of the PukiWiki-Java Connector.

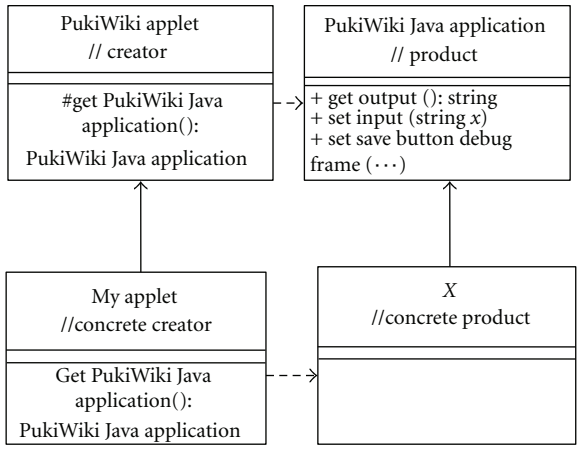

Figure 4: Factory method pattern for the PukiWiki-Java Connector.

#jcon(X) tag. In this mode, the data of X in normal mode wiki page is represented by the text in PukiWiki page syntax in instead of enclosed by the pair of <pre> and </pre> tags. Algorithm 2 shows an example of the source text of the page for editing mode.

The text after <textarea ... > tag until #jcon(X) tag is extracted and saved as the header text for saving.

After that, the getOutput() method of the Java program X is called from the *SaveButtonDebugFrame* class. This method returns the new text data for saving. The new text data is transformed into preformatted text in PukiWiki page format. The new page text for saving is created by concatenating the header text for saving and the new data in pre-formatted text. The new page text is saved to the PukiWiki site using the *Post Method* class of Apache Http Client.

```
[URL of the wiki page]?cmd=edit&page=[page name]
<?php
//PukiWiki - Yet another WikiWikiWeb clone
//
//jcon.inc.php
//     t.yamanoue, 2010
//...
function plugin_jcon_convert()
{
   if (PKWK_READONLY) return '';//Show nothing
   $args = func_get_args();//args
   if (count($args) >= 1) {$aw = array_shift($args);} else {$aw = 'draw';}
   $java_application_name = htmlspecialchars($aw,ENT_QUOTES);
      $ret = '';//return value
   $charset=CONTENT_CHARSET;
   $uri=get_script_uri();
$jcode="application.".$java_application_name.".MyApplet.class";
   $plginname="jcon(".$java_application_name.")";
      $ret = <<<EOD
<div>
<applet codebase= "./javaApplications/bin" code="$jcode"
   archive= "lib/commons-codec-1.3.jar,lib/commons-httpclient-3.1.jar,
         lib/commons-logging-1.1.1.jar"
   width="100" height="100">
<param name="action" value="$uri"/>
<param name="param1" value="plugin=$plginname"/>
<param name="charset" value="$charset"/>
</div>
EOD;
   return $ret;
}
?>
```

Algorithm 1: PHP code for starting the Java program with the PukiWiki-Java Connector.

```
....
<form .... >
<textarea ... >
...
#jcon (X)
Hello! (Data for X in pre-formatted syntax)
</textarea>
</form>
...
```

Algorithm 2: An example of the source text of the page in editing mode.

3. Using Sample Programs

We have created several sample Java programs using the PukiWiki-Java Connector. We also have created a PukiWiki system using these java programs and the plug-in PHP code of Figure 4 in order to allow users to easily try out a combination of Java programs with the PukiWiki-Java Connector and the PukiWiki system. The system can be downloaded at http://yama-linux.cc.kagoshima-u.ac.jp/pukiwiki-java/.

3.1. Simple Text Editor. Creating a simple text editor using the *JTextEditor* class in the swing package of Java is simple. We have embedded a text editor in the PukiWiki system using the PukiWiki-Java Connector. The user can use the text editor by writing the "#jcon(myEditor)" tag at the left side of a line in the editing page of PukiWiki (Figure 5). Then, when the user clicks the update button, the simple text editor and a small window for saving the test editor data will appear on the display (Figure 6). When the user clicks the "save" button in the small window after entering text, as shown in Figure 7, the written letters are saved on the wiki page, as shown in Figure 8.

3.2. Simple Music Editor. Another application is allowing several people to collaborate in creating music. In order to demonstrate this application, we have created a simple music editor for PukiWiki using Java codes on the Web. Figure 9 shows the GUI of the simple music editor. This music editor is started by the "#jcon(musicEditor2)" wiki tag. The user of this editor records a melody using the key board of the editor after clicking the "record" button. A vertical bar, which represents a note of the melody, appears in the editor when a key is pressed while recording. Recording will be ended

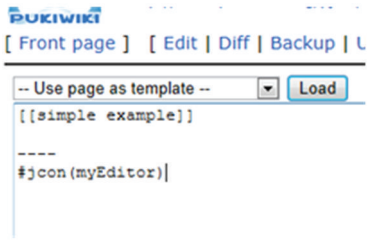

Figure 5: Tag that starts the simple text editor in the editing page of PukiWiki.

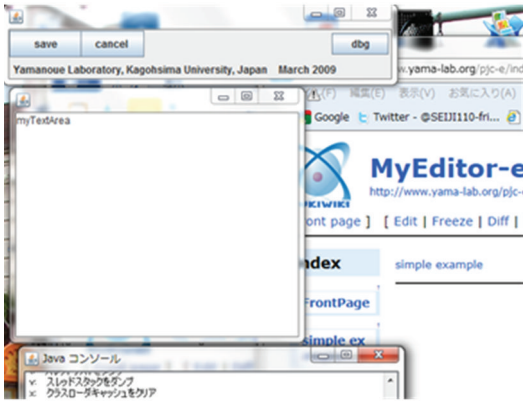

Figure 6: The simple text editor and the small window for saving data.

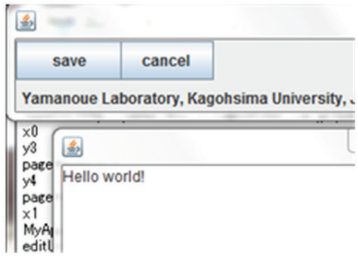

Figure 7: Close-up view of the simple text editor.

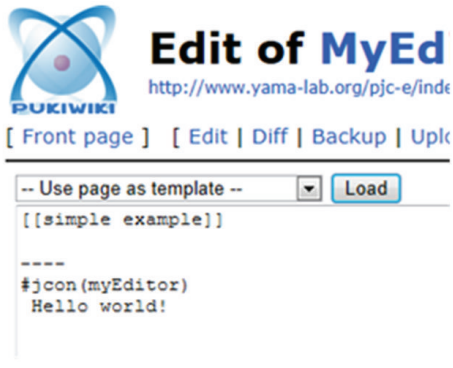

Figure 8: Saved text.

when the "stop" button is clicked. Notes can be added and edited after the recording using "new," "cut," "res" (reset), "mov" (move), "mod" (modify), and "clr" (clear) buttons, and when the "play" button is clicked, the melody is played. The melody is rewind by clicking the "—≪" button. When the "save" button is clicked, the notes are saved in the wiki page. One note of the melody is represented by the following line in the wiki page. *#t,[start time],n,[MIDI note number],l,[duration of the note].*

3.3. Draw. Another application is to share drawings among a group of people who are needed to collaborate frequently. The drawings should be modified by the group members as if the members were drawing on a white board during a meeting. In order to realize this capability, we embedded a drawing program into the PukiWiki system. We modified the drawing program of SOLAR-CATS [8, 9], which is a computer-assisted teaching system. This drawing program is started by the "#jcon(draw)" wiki tag. A drawing of this program is saved to wiki pages as a vector data text. Figure 10 shows an example of a drawing and its saved data. Figure 11 shows an example of using the drawing program to display a map of restaurants. The map was saved on a wiki page and can be modified when a new restaurant is opened or a restaurant closes. We also use this drawing program for in our own classes. Explanatory drawings can be improved during class and saved.

3.4. Programming Environment. Figure 12 shows an example of using a simple programming environment for a basic-like programming language that is embedded in the PukiWiki system. This environment is started by the "#jcon(basic)" wiki tag. The programming environment is also ported from SOLAR-CATS. The drawing program of the previous subsection is used for graphics output. This can be used for situations such like the following.

(1) A teacher of a class shows a program example in a wiki page to students.

(2) Students copy the program to their own wiki page and run the program in a web browser using this programming environment.

(3) The teacher tells students to make a specified program as an exercise of the class by modifying the original program.

(4) The teacher evaluates programs by running the programs at students' pages in the teacher's web browser.

We also have embedded PEN [10], a Japanese programming language's programming environment, into PukiWiki using this API.

Programs on a wiki make wiki pages rich and interactive.

3.5. Voice Recorder. Figure 13 shows an example of using a voice recorder. This is started by the "#jcon(voiceRecorder)" wiki tag. Voice is recorded when the "record" button is clicked. The recorded voice is played when the "play" button is clicked. Recording or playing will be stopped when the

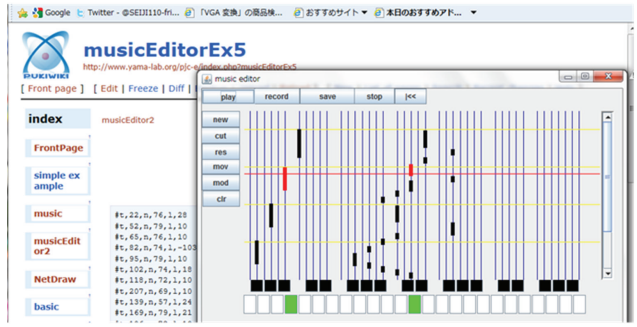

Figure 9: GUI of the simple music editor.

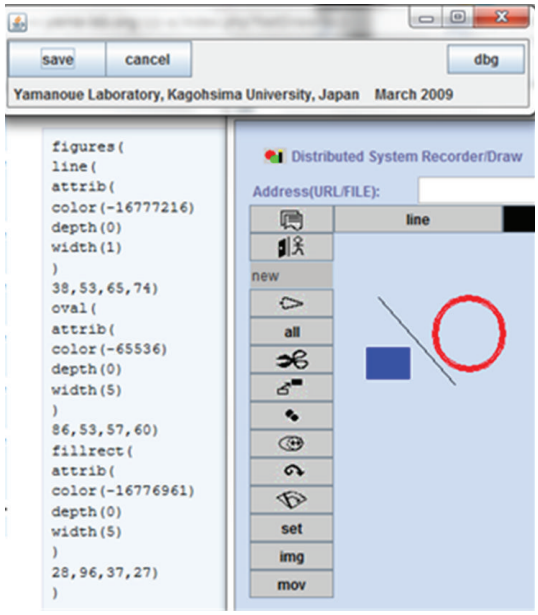

Figure 10: Drawing and its saved data.

"stop" button is clicked. The recorded voice is cleared when the "clear" button is clicked. Wave data of the recorded voice is encoded into a series of letters. A part of them is shown in the recorder's window, and they are stored in the page of PukiWiki when the "save" button is clicked. The saved voice is played again when the "play" button is clicked after the recorder window of the page is shown again.

It is not difficult to save many kinds of binary data to a page of PukiWiki using such encoding like this voice recorder.

4. Embedding Your Own Program

In this section, we demonstrate how to embed your own Java program using the PukiWiki-Java Connector in the PukiWiki system.

4.1. Downloading the API. First, an SDK of the PukiWiki-Java Connector, that is, *javaApplications.zip* or *javaApplications.jar*, should be downloaded to your Java programming environment. These SDKs can be downloaded from http://yama-linux.cc.kagoshima-u.ac.jp/pukiwiki-java/. After unzipping the file, the *javaApplications* directory will appear. In this section, the path of this directory is represented as <javaApplications>.

4.2. Creating the Source Directory. The new Java program is labeled *newAppli*. A new directory, named *newAppli*, should be created in the <javaApplications>/src/application directory. Then, *newAppli* will be the argument of the #jcon PukiWiki plug-in. newAppli is started by the #jcon(newAppli) tag at the left-most position of the editing page of PukiWiki.

4.3. Copying and Modifying MyApplet.Java. Next, MyApplet.java, in the *<javaApplications>/src/application/myEditro* directory, should be copied to the new <javaApplications>/src/application/newAppli directory.

For MyApplet.java in the new directory, the package declaration should be changed to application.new Appli, and the line "frame=new MyEditor();" should be changed to "frame=new New Appli();". Appropriate arguments are added to the constructor if needed. The method getPukiWikiJavaApplication is the factory method. The following is an example of the rewritten MyApplet.java:

```
package application.newAppli;
import connector.*;
public class MyApplet extends PukiWikiApplet{
  public PukiWikiJavaApplication getPukiWikiJava-
  Application(){
    System.out.println(
      "MyApplet.getPukiWikiJavaApplication");
    if(frame==null){
      frame=new MyEditor();
    }
    return this.frame;
  }
}.
```

The "connector" package in the above code includes, for example, a program for communicating with the PukiWiki

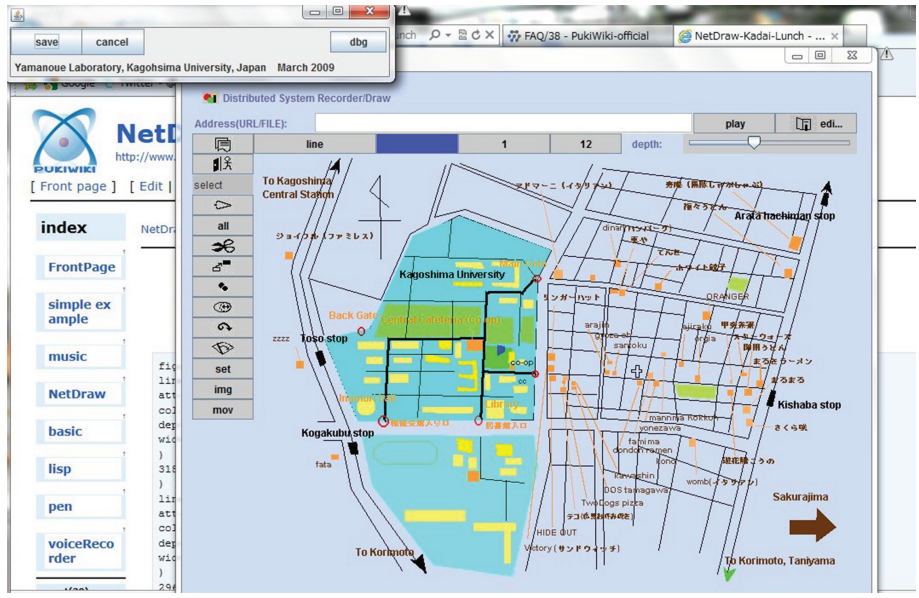

Figure 11: A map of restaurants displayed by the drawing program.

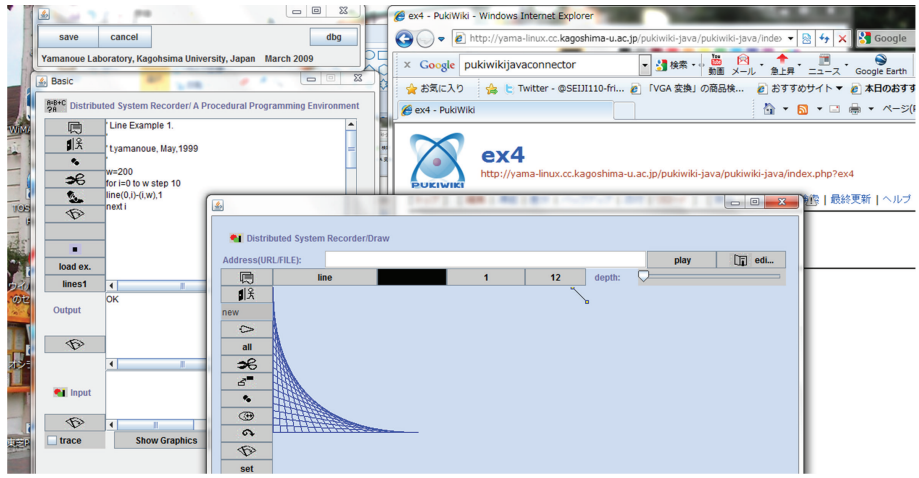

Figure 12: Programming environment for a basic-like programming language.

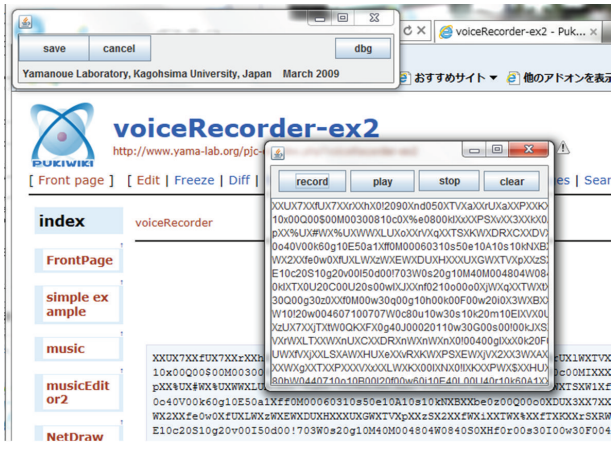

Figure 13: Voice recorder.

system, a wiki page syntax analyzer, and various translators for embedding data in the wiki page.

4.4. Copying the New Application Source Program. The source program of the new application should be copied into the newAppli source directory, that is, <javaApplications>/src/application/newAppli. This application must have a subclass of the JFrame class. This subclass is started from the PukiWiki system. The name of this class is herein assumed to be NewAppli, and the source of this class is NewAppli.java. Other source programs are also copied into an appropriate directory if these programs are required by NewAppli.java.

4.5. Implementing the Interface. NewAppli.java must implement the PukiWikiJavaApplication interface, as shown in

Section 2. The connector package must also be imported, as exemplified by the following code:

```
package application.newAppli;
import connector.*;
...
public class NewAppli extends JFrame
   implements PukiWikiJavaApplication
{
      ...
}.
```

The following three methods should be overridden in order to implement the PukiWikiJavaApplication interface, as shown in Section 2.

```
@Override
public String getOutput() {
      //TODO Auto-generated method stub
}
@Override
public void setInput(String x) {
      //TODO Auto-generated method stub
}
@Override
Public void setSaveButtonDebugFrame
   (SaveButtonDebugFrame f) {
      //TODO Auto-generated method stub
}.
```

The following code shows an example of an implementation of the overridden methods.

```
public String getOutput() {
      return this.myTextArea.getText(); //add
}
public void setInput(String x) {
      this.myTextArea.setText(x); //add
}
public void setSaveButtonDebugFrame
         (SaveButtonDebugFrame f) {
}.
```

4.6. Compiling and Uploading. The source codes in the newAppli directory should then be compiled. We assume that the compiled classes in the newAppli directory are in <javaApplications>/bin/application/newAppli directory and all other classes for running the NewAppli are in the <javaApplications>/bin directory. A local server program such as Xampp [11] can be used for debugging on the local host. If there are no errors, then all classes should be uploaded into the *./javaApplications/bin* directory of the PukiWiki system of the web server. The PukiWiki-Java Connector requires Apache httpclient. The jar files of httpclients must be in the*./javaApplications/bin/lib* directory. Finally, adding or modifying the PukiWiki system is not necessary.

5. Evaluation

5.1. Enriched Wiki Content. In Section 3, we demonstrated that a draw application, programming environments, and a music editor can be embedded into wiki content. Introducing new applications with the PukiWiki-Java connector is simple.

5.2. Minimized Porting Cost. The PukiWiki-Java Connector is already equipped with a wiki page syntax analyzer and various translators for embedding data into a wiki page. These components are easy to reuse by a programmer using the factory method pattern without thinking of such cumbersome things like a syntax analyzer and translators. Approximately three days were needed for one person to embed the drawing program of SOLAR-CATS into the PukiWiki system. Approximately three weeks were needed for one person to create a similar drawing program from scratch. Approximately two hours were needed for one person to embed the simple text editor into the PukiWiki system. Finally, approximately one day was needed for one person to embed the simple music editor and the programming environment for a basic-like programming language.

6. Related Research

There are many kinds of wiki software. Some of them are capable to have rich media content like PukiWiki with PukiWiki-Java connector. We compare such wikis and related work with PukiWiki with PukiWiki-Java connector in this section.

6.1. Java Applet Extensions and Plug-Ins. Some wikis, such as MediaWiki [12] and PukiWiki [3], have an extension or plug-in for starting a Java applet. However, these extensions do not have the ability to save the applet's data in a wiki page.

6.2. AnyWikiDraw. AnyWikiDraw [13] data can be saved on a wiki page. In addition, AnyWikiDraw can be used with various wikis, such as TWiki, PmWiki, and MediaWiki. However, AnyWikiDraw is a specialized system for drawing. On the other hand, Pukiwiki-Java connector is a general purpose API for embedding Java programs.

6.3. Galaxy Wiki. Galaxy Wiki [14] is a software development environment. Developers are able to experience "writing wiki page is wring source code" using this. Moreover, developers are able to compile, execute, and debug programs in wiki pages too. Galaxy Wiki is also a specialized system for software development.

6.4. AddesoWiki. Adessowiki [15] is a collaborative environment for development, documentation, teaching and knowledge repository of scientific computing algorithms. The system is composed of a collection of collaborative web pages in the form of a wiki. The articles of this wiki can embed programming code that will be executed on the server when the page is rendered, incorporating the results as figures, texts, and tables on the document. On the other hand, programs which use PukiWiki-Java connector are executed on local hosts. This makes the CPU load of hosts lighten.

6.5. Lively Wiki. Lively Wiki [16] is a development and collaboration environment based on the Lively Kernel, which enables users to create rich, interactive Web pages and applications. Lively Wiki can be used to expand itself. It is written entirely in JavaScript. Although our goals are similar to those of Lively Wiki, for conventional users, who cannot write JavaScript codes, using Lively Wiki is too complicated.

6.6. mbed. mbed [4] is a tool for rapid prototyping with microcontrollers. A web top programming environment can be used in the Web site of mbed such as Galaxy Wiki and AddesoWiki. The programming environment of mbed is also a specialized system for software development.

6.7. WIKIAPI. The WIKIAPI [2] provides a standard Java API to enable reading and writing to a wiki from within a Java Application. The Wiki API supports MediaWiki, MoinMoin Wiki, TWiki, and Confluence Wiki. WIKIAPI is used for accessing wiki pages from a Java application running on a local host. On the other hand, PukiWiki-Java connector is used for embedding Java applications and their data on wiki pages.

7. Concluding Remarks

We proposed the PukiWiki-Java Connecter, which is an API of applets for easy, unified data input and output at a remote host. The PukiWiki-Java Connecter is used to save Java program data in wiki pages. We also combined the proposed API and a wiki system by introducing a wiki tag for starting Java applets. A number of Java programs were demonstrated to embed into the wiki system in a short time. We intend to improve the proposed API by facilitating its use after obtaining feedback from users.

References

[1] W. Cunningham, "Wiki Wiki Web," 1995, http://c2.com/cgi/wiki?WikiWikiWeb.
[2] WIKIAPI, 2012, http://jwikiapi.sourceforge.net/.
[3] PukiWiki, 2001, http://pukiwiki.sourceforge.jp/.
[4] mbed, 2012, http://mbed.org.
[5] J. Franks, P. Hallam-Baker, J. Hostetler et al., "HTTP Authentication: Basic and Digest Access Authenti," RFC 2617, 1999.
[6] E. Gamma, R. Helman, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley Longman, Boston, Mass, USA, 1995.
[7] HttpClient Home, 2010, http://hc.apache.org/httpcomponents-client-ga/.
[8] T. Yamanoue, "Sharing the same operation with a large number of users using P2P," in *3rd International Conference on Information Technology and Applications (ICITA '05)*, vol. 2, pp. 85–88, July 2005.
[9] T. Yamanoue, "A casual teaching tool for large size computer laboratories and small size seminar classes," in *Proceedings of the ACM SIGUCCS Fall Conference (SIGUCCS '09)*, pp. 211–216, October 2009.
[10] T. Nishida, A. Harada, R. Nakamura, Y. Miyamoto, and T. Matsuura, "Implementation and evaluation of PEN: the programming environment for novice," *IPSJ Journal*, vol. 47, no. 4, pp. 1063–1076, 2006 (Japanese).
[11] Wikipedia, 2011, http://www.wikipedia.org/.
[12] MediaWiki, 2007, http://www.mediawiki.org/wiki/MediaWiki.
[13] AnyWikiDraw, 2011, http://www.randelshofer.ch/anywikidraw/index.html.
[14] W. Xiao, C. Chi, and M. Yang, "On-line collaborative software development via wiki," in *International Conference on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA '07)*, pp. 177–183, October 2007.
[15] R. A. Lotufo, R. C. Machado, A. Körbes, and R. G. Ramos, "Adessowiki on-line collaborative scientific programming platform," in *5th International Symposium on Wikis and Open Collaboration (WiKiSym '09)*, October 2009.
[16] R. Krahn, D. Ingalls, R. Hirschfeld, J. Lincke, and K. Palacz, "Lively wiki a development environment for creating and sharing active web content," in *5th International Symposium on Wikis and Open Collaboration (WiKiSym '09)*, Orland, Fla, USA, October 2009.