New Usage of Web Applications—Web Application Convergence Assisted by Wrappers

Yuusuke Nakano[†], *Yoji Yamato, Michiharu Takemoto, and Hiroshi Sunaga*

Abstract

This article describes the mechanism by which wrappers work and a method that supports wrapper generation. It also introduces some examples of the utility of wrappers. Wrappers, which enable ordinary web applications to be used as components, let us make new applications by converging the components.

1. Web application convergence and wrappers

Many people use various web applications. Some people may use a hotel search web application and a route plan web application to find a route from their home to a hotel that they have found and print out the results. In this way, several web applications and other systems such as printers are typically used at the same time. Useful applications can be made by converging web applications and other systems. This web application convergence requires each application to be a component. Our wrappers enable such web applications to be used as components.

2. Importance of components

In NTT Network Service Systems Laboratories, we conduct research on service platforms to provide ubiquitous services by combining various components. Users receive appropriate ubiquitous services from the platform when components that are suitable for each user's situation are combined dynamically. Providing such ubiquitous services requires various components to meet changes in a user's situation. Those components are combined to make a service more suitable for handling the situation.

Mashups are becoming popular these days. A mashup is used to create Web 2.0 applications. Mashups stimulate user-initiative in the creation of applications because of their simplicity. In using a Mashup, application creators find components such as web services and converge them. In this way, they can create applications easily, but a huge number of components is necessary to enable them to create applications as they like.

3. Making components from web applications

3.1 Outline

We have conducted research about wrappers that make web applications, especially web applications that perform searches such as hotel search web applications, usable as components. The wrapper interconverts a web application's protocol and a component's protocol, which is SOAP (simple object access protocol), and makes the web application usable as a web service. However, creating a wrapper for each web application is laborious. Therefore, our wrapper interconverts in accordance with a configuration file written for each web application (Wrapper in **Fig. 1**).

† NTT Network Service Systems Laboratories
Musashino-shi, 180-8585 Japan

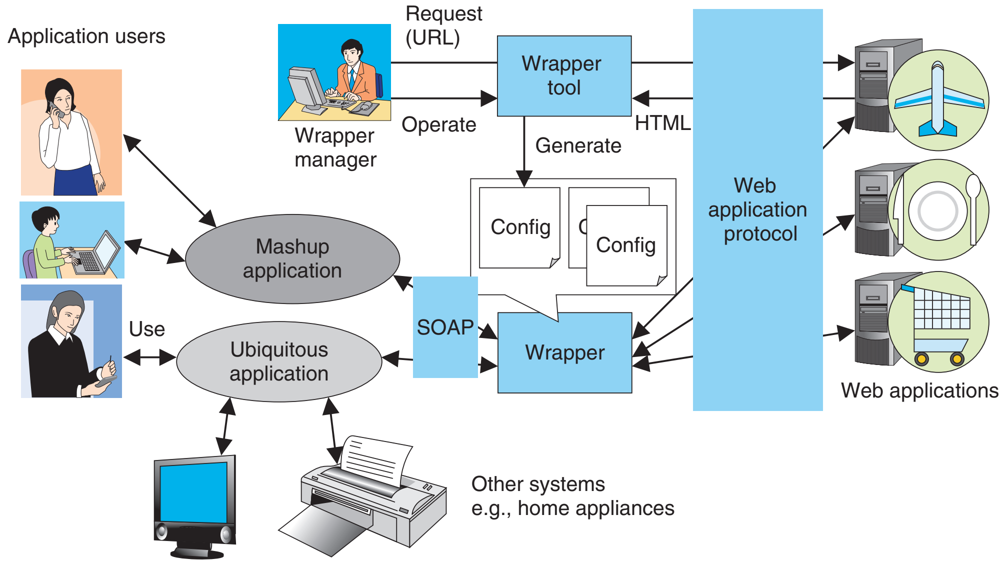


Fig. 1. Overview of wrapper and wrapper tool.

For example, a configuration file for a hotel search web application has to be written to wrap the web application and make it usable as a hotel search component. To make a huge number of web applications usable as components, a huge number of configuration files must be written, so a method of reducing this work is required.

3.2 Generation of configuration files by automatic extraction of search results

A configuration file contains descriptions of where to extract from an HTML (hypertext markup language) document given by a web application and how to convert the extracted search results into an XML (extensible markup language) document. Our wrapper-generation tool supports the generation of configuration files by automatic extraction of search results from the HTML document and the writing of the position of the extracted search results in the HTML document into the configuration file (Wrapper tool in Fig. 1).

To extract the search results, the wrapper tool uses a feature of an HTML document generated by a search web application as below. Most search web applications generate HTML documents automatically using a database. Such HTML documents tend to have a repetitive tag pattern. The wrapper tool extracts search results from the HTML document automatically by using the repetitive pattern.

4. Utilization of repetitive depth pattern of tag

Our method of extracting search results finds areas in an HTML document that have a repetitive depth pattern of tags to find the repetitive tag pattern. The depth of tags means the number of nesting levels and is visualized as shown in **Fig. 2(a)**. HTML documents often have the repetitive depth pattern shown in **Fig. 2(b)** because web applications generate HTML documents mechanically. The wrapper tool extracts the area that has the same repetitive depth pattern as the search results.

This method is similar to a method used by a human being to find search results in an HTML document generated by a web application. When search results need to be found in an HTML document written in a foreign language, an area that has a repetitive pattern, such as a table or list, is found. The person finds the correct search results even if there are many other elements such as menus and banners. In this case, the person will use neither the HTML tags nor the meaning of text in the HTML document but the appearance of the HTML document to find the results of the hotel search. In this way, the wrapper tool extracts search results from various HTML documents generated by web applications by using a method used by human beings in their daily lives.

5. Experimental results

We collected 100 HTML documents written in

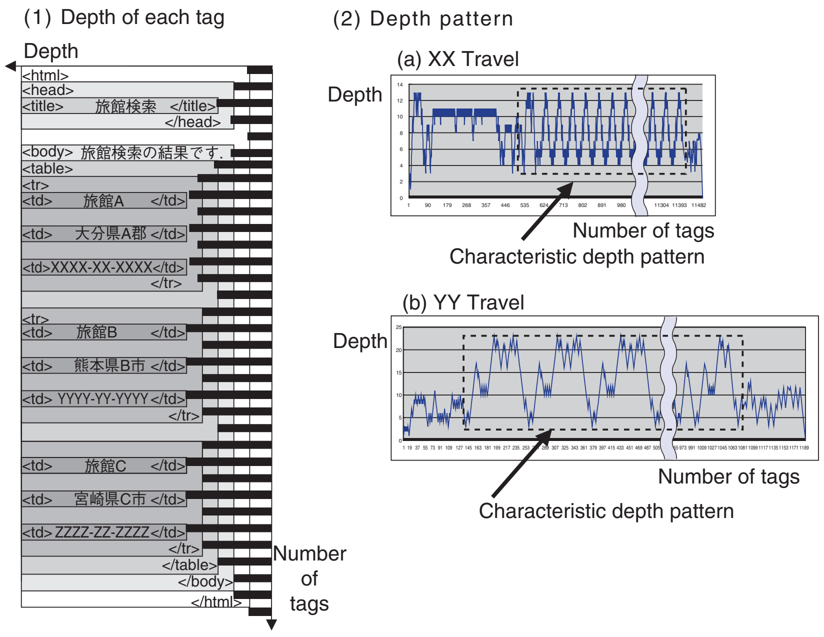
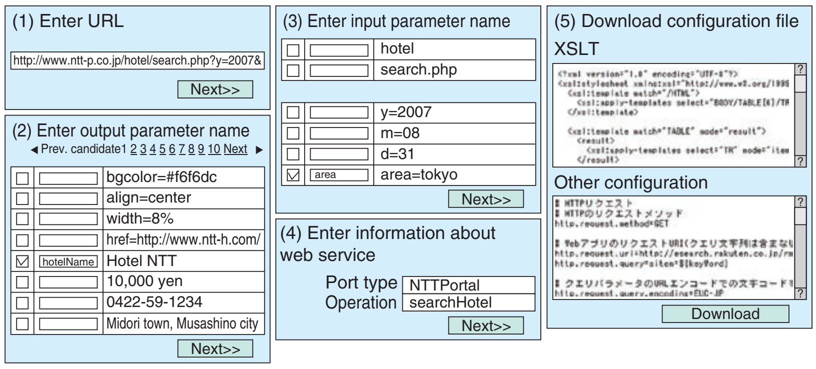


Fig. 2. Depth of tags.


(1) Enter URL

http://www.ntt-p.co.jp/hotel/search.php?y=2007&

Next>>

(2) Enter output parameter name

◀ Prev. candidate1 2 3 4 5 6 7 8 9 10 Next ▶

☐		bgcolor=#f6f6dc
☐		align=center
☐		width=8%
☐		href=http://www.ntt-h.com/
☑	hotelName	Hotel NTT
☐		10,000 yen
☐		0422-59-1234
☐		Midori town, Musashino city

Next>>

(3) Enter input parameter name

☐		hotel
☐		search.php
☐		y=2007
☐		m=08
☐		d=31
☑	area	area=tokyo

Next>>

(4) Enter information about web service

Port type	NTTPortal
Operation	searchHotel

Next>>

(5) Download configuration file

XSLT

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999
  <xsl:template match="/HTML">
    <xsl:apply-templates select="BODY/TABLE[6]/TR
  </xsl:template>

  <xsl:template match="TABLE" mode="result">
    <result>
      <xsl:apply-templates select="TR" mode="item
    </result>
```

Other configuration

```
# HTTPリクエスト
# HTTPのリクエストメソッド
http.request.method=GET

# WebアプリのリクエストURI(クエリ文字列は含まない
http.request.uri=http://esearch.rakuten.co.jp/rm
http.request.query=sitem=${keyWord}

# クエリパラメータのURLエンコードでの文字コード
http.request.query.encoding=EUC-JP
```

Download

Fig. 3. GUI of wrapper tool.

Japanese generated by 100 web applications providing services, such as an online shop, hotel search service, and rental apartment search service. At the same time, we implemented an experimental program that finds an area that has the depth pattern and extracts a segment around the area as a search result segment.

Then, we tried to extract search result segments from the HTML documents using the experimental program and found that the program extracted the correct result segments from 80% of the HTML documents. This means that our method can process the HTML documents generated by most web applications.

One of the main reasons for extraction failures was that there were other non-search result segments that had repetitive depth patterns. For example, collections of links and menu segments often had the repetitive tag pattern, and they had the repetitive depth pattern. The experimental program could not determine whether the found segment was the correct search result segment.

6. Wrapper tool

We implemented the wrapper tool based on the experimental program. The tool generated the configuration files for the wrapper. Its graphical user interface (GUI) is shown in **Fig. 3**. Users operated the tool and made the configuration files as below.

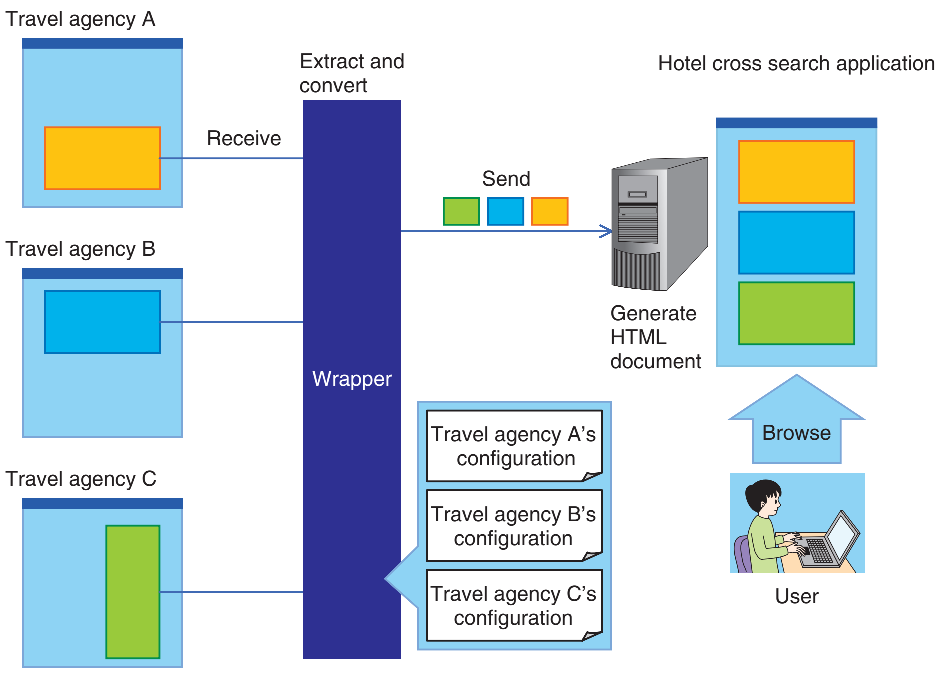

Travel agency A
Travel agency B
Travel agency C
Receive
Extract and convert
Wrapper
Send
Hotel cross search application
Generate HTML document
Travel agency A's configuration
Travel agency B's configuration
Travel agency C's configuration
Browse
User

Fig. 4. Example of wrapper application.

(1) The user entered a uniform resource locator (URL) of a search result HTML document generated by the target web application and clicked the *next* button on the tool's GUI.

(2) The wrapper tool showed a search result such as information about a hotel. The user confirmed that the information was the correct search result; otherwise, the user was shown the next search result candidate. After being shown the correct search result, the user checked the check boxes on the GUI to indicate output values of the web application used for output values of the component, which is a web service, and entered their output parameter name. For example, the user checked the box next to hotel name and entered *hotelName* as an output parameter name.

(3) The wrapper tool showed elements in the URL of the search result HTML document. These elements are directory names and a file name in the URL path and input values in the URL query string. The user checked boxes to indicate input values of the web application used for input values of the component and entered their input parameter name. For example, the user checked the box next to an area name and entered *area* as an input parameter name.

(4) The user entered the port type, which is a class name, and the operation, which is a method name.

(5) The user confirmed whether the generated XSLT file, which has conversion rules for conversion from HTML to SOAP, and another configuration file were correct. These two files were downloaded as a zipped file.

In this way, the user obtained the configuration files through an easy operation, and those files make the web application usable as a component by inputting the configuration files into the wrapper.

7. Application of wrapper technology

Are wrappers useful only for ubiquitous services or mashup services? One of the most popular uses is a bank account aggregation service or cross search services. These services show the search results of multiple web applications on one page. Users can compare those search results by looking at the page. To make such services, the service creator must write programs for each web application to extract important segments from HTML documents generated by the web applications.

An example application for a hotel cross search is shown in **Fig. 4**. In this case, the application shows search results for travel agencies A, B, and C. The wrapper extracts search result segments from the HTML documents generated by each web application, converts them into a compatible format, and sends them to a server for the cross search application. The server generates an HTML document from the received search results and provides it to the user. The service creator creates cross search applications easily and with less programming by letting the wrapper extract important segments and by using the wrapper tool. In addition, when the design of HTML documents generated by a web application changes, the cross search application manager generates a new configuration file easily by using the wrapper tool, and he/she does not have to write a program. This means that the wrapper and the wrapper tool can also reduce the maintenance cost of cross search applications.

8. Conclusion

We implemented wrappers and a wrapper tool that can make a huge number of web applications easily usable as components. As future work, we plan to develop wrappers to reduce maintenance costs more effectively by following changes in web application design automatically.

Yuusuke Nakano

Researcher, Emerging Communication Architecture Project, NTT Network Service Systems Laboratories.

He received the M.E. degree in systems engineering from Wakayama University, Wakayama, in 2005. He joined NTT Laboratories in 2005. There, he has been engaged in developmental research on ubiquitous network systems. He is interested in ubiquitous computing environments and 3D virtual worlds. He is a member of the Information Processing Society of Japan (IPSJ).

Yoji Yamato

Researcher, Emerging Communication Architecture Project, NTT Network Service Systems Laboratories.

He received the B.S. and M.S. degrees in science from the University of Tokyo, Tokyo, in 2000 and 2002, respectively. He joined NTT Laboratories in 2002. There, he has been engaged in developmental research of peer-to-peer network systems, ubiquitous network systems, and service delivery platforms. During 2005–2007, he was a secretary of the Next Generation Networks Software Technical Committee of the Institute of Electronics, Information and Communication Engineers (IEICE) of Japan. He received the Next Generation Networks Software Young Researcher Award from IEICE Next Generation Networks Software Technical Committee in 2007 and the Communications Society's Distinguished Contributions Award from IEICE in 2007. He is a member of IEICE and IPSJ.

Michiharu Takemoto

Senior Research Engineer, Ubiquitous Service Systems Laboratory, NTT Network Innovation Laboratories.

He received the B.S. and M.S. degrees in information science from the University of Tokyo, Tokyo, in 1992 and 1994, respectively. He joined NTT Network Service Systems Laboratories in 1992. Since he joined NTT, his research interests have included the distributed computing environment and its applications. He received the Young Investigators Award from IEICE in 1998. During 1999–2000, he was a visiting scientist at the Laboratory for Computer Science, Massachusetts Institute of Technology. He is a member of IEEE, IPSJ, and IEICE.

Hiroshi Sunaga

Executive Manager, Ubiquitous Service Systems Laboratory, NTT Network Innovation Laboratories.

He received the B.E. and M.E. degrees in control engineering from Tokyo Institute of Technology, Tokyo, in 1981 and 1983, respectively. He received the Ph.D. degree in information sciences from Tohoku University, Miyagi, in 2002. Since joining NTT in 1983, he has been engaged in R&D of telecommunication software for switching nodes (packet, telephone, PHS, ATM, VoIP, and mobile), realtime operating systems, and P2P/grid systems. His current research interests include ubiquitous services and network service platform systems. He has been active in the standardization of OSI communications, network management, and VoIP in ITU-T and TTC. He is a member of IEEE.