Using IntelligentPad as a Server-Side Technology

Takahiro Endo¹, Yuzuru Tanaka²

¹Innovation Plaza Hokkaido
Japan Science and Technology Agency
Nishi 11, Kita 19, Kita-ku, Sapporo
0600819 Japan
endo@meme.hokudai.ac.jp

²Meme Media Laboratory
Hokkaido University
Nishi 8, Kita 13, Kita-ku, Sapporo
0600828 Japan
tanaka@meme.hokudai.ac.jp

Abstract: This paper proposes a framework to utilize the IntelligentPad system as Web Services so as to expand the application range of our knowledge media technology. We have ever developed applications of the IntelligentPad which enable wrapping, re-editing and composing of various resources on the Web. However, to use such applications, users have to execute the IntelligentPad system on their own computers or flatten the pad applications into HTML data to use them as Web applications, and that becomes a barrier of the spread of the IntelligentPad. To facilitate the use of the IntelligentPad, we require the introduction of generic Web technologies into the IntelligentPad system in order to enable access to pad applications from generic clients such as Web browsers. In our framework, we define Web Services operations for the IntelligentPad system and implement a server for Web Services. Also, we introduce a framework to communicate between pad applications and mobile phone applications via SOAP protocol so as to expand the application range of the IntelligentPad system across mobile computing or ubiquitous computing.

1 Introduction

In the current World Wide Web, most Web pages are designed to be human-readable and to be operated by user interactions. Different Web sites have different formats for Web pages and different interfaces for Web applications. It makes difficult to extract a portion of Web pages and to integrate Web applications programmatically. For that reason, Web pages and Web applications cannot be linked as though they are interoperable and reusable software components.
Recently, a new concept ‘Web 2.0’ is emerging around the World Wide Web as a collective term of next generation Web technologies, for instance Semantic Web \cite{BHL01}, Weblog (or blog), RSS or Web Services \cite{Bo04}. In the inception of the Web, Web documents have only static content and are dealt as the minimum element to make links though we can embed anchors into documents, and Web sites provide a set of such documents. Next, Web documents include dynamic content by using Server Side Include (SSI), JavaServer Pages, Active Server Pages, PHP and so on. Nowadays contents of Web sites are managed and dynamically generated by server-side technologies such as Content Management Systems, while each page is composed of articles and the articles become the minimum element to make permanent links or permalinks. Weblog system provides another link mechanism named trackback and RDF provides the mechanism to add metadata onto Web pages. Now, we have many kinds of tools to enhance usability of the Web. In the target of Web 2.0, users can produce new resources or customized applications by combining or remixing fragments of Web resources for individual purpose. Therefore, we can regard the ‘Web as Platform’.

The knowledge media technology IntelligentPad \cite{TI89} \cite{Ta92} enables heterogeneous media to be handled as homogeneous software components, that is called a pad. Pads can be linked together with a simple drag-n-drop operation and can communicate with each other by a unified message protocol. We have ever introduced the technique for clipping a portion from any Web page or Web application to make the portion as a pad and linking such pads together to build a new integrated tool as a pad application \cite{Fu04} \cite{IT03}.

On the other hand, the executable program of the IntelligentPad system is necessary to use pad applications, even new components can be created by clipping from the Web. For expanding the application range of the IntelligentPad, pad applications should be accessed through more generic technologies which are widely used in the Web. In our approach, the IntelligentPad system works as a Web server and the pad applications are utilized as Web Services from external clients written in Java, Perl or the like. Thus, many kinds of Web resources are dynamically combined with pads and pad applications can be widely reused in the Web.

Also, users demand the improvement of usability for the Web, not only for desktop environment but also the mobile computing environment for using mobile-phones and PDAs. The mobile-phones have some restrictions about the display capability or the access to the Internet, but the latest mobile-phones, or smart phones, equip WiFi function and full-featured Web browser software almost equal to the Web browsers for desktop, in addition to phone-call function and Java runtime environment. The users of such mobile devices require more sophisticated and rich applications rather than old-fashioned applications. We focus on these aspects to develop the architecture for accessing Web Services from the mobile devices by applying the knowledge media technology.
2 IntelligentPad Web Services

At first, we define operations to make use of the IntelligentPad system as a Web service and describe the definition of the operations in a file written in Web Services Description Language (WSDL) [Mi03]. Clients of the Web services interpret the WSDL to make requests and process responses.

Web Service providers offer various operations to serve a variety of functions and define the WSDL at each service. The operations of the Web services are combined to build a target application.

Though pad applications offer different functions, the IntelligentPad Web Services has only a few common operations for the different pad applications, because the message protocol of the IntelligentPad is very simple where pads use only three messages, that are ‘set’, ‘gimme’ and ‘update’. Those operations consist of the operations for pad instance and the operations for pad message.

2.1 Operations

We define following operations for the IntelligentPad Web Services (IPWS) as a basis. They are minimal set to utilize functionality of pad applications through the IPWS from external clients. The clients combine these operations to build their own applications.

2.1.1 Operation for pad instance

Operations for pad instance are used to retrieve information related to pad instance.

(1) An operation `getPadList()` is to obtain an array of names of pads registered in the IPWS server.

(2) An operation `getPadNameOf(string padname)` is to obtain information of the pad specified by `padname`. The information stored in a structure data contains a class name of the pad, an ID No. and so on.

(3) An operation `getPadXml(string padname)` is to obtain a string describing storage format data of the pad written in XML specified by `padname`.

2.1.2 Operations for pad message

Operations for pad message are used to send messages to pad instances from the IPWS server indirectly. If the IPWS server receives a request of these operations, the IPWS server sends a pad message corresponding to the requested operation to the actual pad instance.

(1) An operation `invokeSet(string padname, string slotname, object value)` is to send a Set message to the pad specified by `padname` for the `slotname` slot with a `value`, and return a boolean value as a result of the Set message.
(2) An operation `invokeGimme(string padname, string slotname)` is to send a Gimme message to the pad specified by `padname` for the `slotname` slot and return a value of any object as a result of the Gimme message.

(3) An operation `invokeUpdate(string padname)` is to send a Update message to the pad specified by `padname` and return a boolean value as a result of the Update message.

(4) An operation `invokeSetAndGimme(string padname, string slot4set, object value, string slot4gimme)` is to send both a Set message and a Gimme message sequentially in one operation. This is a utility operation to get a value as a result of the Set message.

2.1.3 SOAP Message

The IntelligentPad Web Services uses SOAP [Mi03] as a messaging protocol. SOAP is a standard for exchanging XML-based messages over the network. The IPWS uses SOAP messaging as Remote Procedure Call (RPC), where a client sends a request message to the IPWS server, and the server immediately sends a response message to the client.

2.3 Server implementation

For utilizing the IntelligentPad system as Web Services, there needs a server program to receive, interpret and send SOAP messages. Because the HTTP protocol is widely used for a transport protocol in SOAP, we implement a Web server and a SOAP processor into the IntelligentPad system. This Web server is named IntelligentPad SOAP server (IPSS). The IPSS processes normal HTTP requests and SOAP requests, and also manages pad instances. The instances are managed by registering into a dictionary variable of the IPSS and clients can specify a pad instance by its name as key.

The client of the IPSS specifies the requesting service, e.g. file transfer or SOAP, by a first path part of the request URL. The typical URL is

```
http://hostname:port_number/service_type/optional_parameters
```

where the `service_type` should be `soap`, `iapp`, `rss` or `rest`.

When the `service_type` is `soap`, a message of an HTTP request is handled as a SOAP request. The IPSS interprets the SOAP request message sent from a client to extract an operation name and operation parameters, executes the operation by communicating with the IntelligentPad system, and then replies a SOAP response message to the client. If any errors occur in the executing process, a fault response message will be returned to the client.

When the `service_type` is `iapp`, the IPSS executes file transfer process related to mobile phone applications. Normally, Java program files and data files related to the applications will be transferred.
When the `service_type` is ‘rss’, the IPSS generate RSS data for pad instances registered in the IPSS server to describe the information of slots of each pad. Once a client knows the slot information of the pad, the client can access any slot via SOAP operations.

### 3 Web application to Web Service

Here, we show an example of the composition of Web applications: one is a wireless device location service, which provides connection status information at each access point of a wireless LAN system, and another is a Web camera service, which provides photographic images taken by remote cameras placed at the location of the access points. Later, we explain how to access the composed application from a SOAP client implemented on a mobile phone.

#### 3.1 Wireless Device Location Service

The wireless device location service is provided from a Web server as a Web application with REST API. Input parameter includes device name, location name, type of event, and so on. The service outputs the list of data, including type of event, date, location name and device name, described in RSS format. A sample URL for request is below.

```http
http://wdls_server/search/rss.rb?hostname=topica&eid=enter+connect+lost+leave&rid=4F+3F1+3F2+2F+1F&limit=20
```

The parameter `hostname` designates the name of a wireless device to be searched, `eid` specifies the type of events, `rid` specifies the locations, and `limit` specifies the number of items to be included in a result.

A pad application for the wireless device location service, named `LocationServicePad`, consists of four primitive pads (Figure 1).

1. A `URLGeneratorPad` sets up search parameters for REST API and generates a URL
2. An `RSSClientPad` fetches RSS data from the service at the URL
3. An `XmlViewPad` processes XML data and extract an element by XPath
4. An `RSSItemPad` processes an item node in RSS and access its child nodes

The `URLGeneratorPad` is composed of text fields, check-boxes, popup-menus and `StringConcatenatorPads` to set up the input parameters for the service and generate an URL into a ‘string’ slot. The ‘string’ slot is connected to a ‘url’ slot of the `RSSClientPad`.

When the `RSSClientPad` receives a new URL string to the ‘url’ slot, the pad sends an http request to the wireless device location service to get RSS data. An XML document object is generated from the RSS data to be stored into an ‘rss’ slot. The ‘rss’ slot is connected to a ‘baseElement’ slot of the `XmlViewPad`. 

182
The XmlViewPad stores the XML document in the ‘baseElement’ slot. When an XPath expression string is set to an ‘XPath’ slot, the XmlViewPad evaluates the XPath expression to extract an XMLNodeList object from the XML document and stores it in a ‘SelectedNode’ slot.

When the RSSItemPad receives the XMLNodeList object for an ‘RssItem’ slot, a first item in the XMLNodeList, that is an XMLNode object, is stored into the ‘RssItem’ slot. At same time, the names of the child nodes of the XMLNode object are automatically registered as slots of the RSSItemPad to access the value of the child nodes. In the RSS of the location service, the child nodes consist of title, link, description and date elements. Also, the RSSItemPad has some StringPads to display the value of the title slot, the link slot, the description slot and the date slot.

3.2 Web Camera Service

The Web camera service serves photographic images taken by cameras located at the access points for wireless devices. The service is a simple Web application. When receiving an HTTP request with a URL of an image file, the service returns the image data.

A WebCameraPad stores the list of URLs corresponding to the images at each access point location, and is composed of a PopUpMenuPad to select a location from a pop-up menu and an ImageViewPad to retrieve and display image data from the Web camera service specified with the URL which is set to the ‘url’ slot.
3.3 Composition of Web applications

The linkage between the LocationServicePad and the WebCameraPad is executed as follows (Figure 2). The LocationServicePad extracts the location of the specified device from the RSS data and stores it into the ‘description’ slot of the RSSItemPad. The value of the ‘description’ slot is shown in the StringPad connected to the slot, then a shared copy of the StringPad is created and pasted onto the PopUpMenuPad of the WebCameraPad. The StringPad is connected to the ‘SelecteKey’ slot of the PopUpMenuPad. Once the search condition of the LocationServicePad is input to retrieve the latest location information, the slot value of the StringPad displaying the location is also changed and the value is propagated to its shared copy pad to input to the WebCameraPad. As a result, the WebCameraPad displays the photographic image of the location where the wireless device exists. Thus the location service and the Web camera service are integrated dynamically without coding any program.

4 Web Service and Mobile Phone

The IntelligentPad Web Services provides a new framework to solve some problems concerning mobile phone applications and enhances the usability of the mobile phones.

4.1 Problems of Mobile Phone applications

We adopt the ‘i-appli’ architecture produced by NTT DoCoMo as an implementation platform. The ‘i-appli’ is a kind of mobile phone application which is developed and executed in Java environment. The profile of the ‘i-appli’ is defined as ‘DoJa’ [NT05].

Usually, mobile phone applications have a few restrictions to access the Internet. The ‘i-appli’ can only communicate with a single server from which the ‘i-appli’ has downloaded through the gateway of the mobile phone operator. The communication protocol is limited to HTTP or HTTPS and also the number of connections to the server is limited to only one at once.
Because the ‘i-appli’ is executed under single-task environment, a user must terminate the currently running application to activate another application if the user wants to use several applications alternately. And the applications cannot communicate with each other directly and share their data easily. Under such environment, it is impossible to dynamically combine and link multiple applications. We therefore try to solve these problems by composing a pad application of several Web applications and accessing the integrated pad application from a mobile phone application via SOAP.

4.2 Implementation of SOAP client for Mobile Phone

The IPWS can be used from generic SOAP clients, because the IPSS implements the standards of Web services. We develop a SOAP client, which works on mobile phones under Java runtime environment, as an ‘i-appli’ for the IPWS.

To use the ‘i-appli’ application named ‘iLocationService’, which accesses both the LocationServicePad and the WebCameraServicePad, the application is first downloaded from the IPSS while the application stores the URL of the IPSS as a parameter. Once the download is completed, the application is stored into user’s mobile phone (Figure 3). When a user activates the application, an initial screen appears to input search conditions for the location service. Once the user operates the screen and depresses a button, the application starts to send the sequence of SOAP requests to the IPSS. The ‘iLocationService’ application sends SOAP requests in following order.

(1) invokeSet(RSSClientPad, ‘url’, aUrl)

The application generates aUrl for input to the Device Location Service Server (DLSS). The IPSS and the DLSS are executed at different servers. And aUrl is set to the ‘url’ slot of the RSSClientPad. Consequently, the RSSClientPad retrieves RSS data from the DLSS and also the RSS data is set to the ‘baseElement’ slot of the XMLViewPad via slot connection. After finishing pad operations, the IPSS replies a SOAP response to the ‘iLocationService’.

(2) invokeSetAndGimme(XMLViewPad, ‘xpath’, anXPath, ‘valueOfNode’)

Next, the ‘iLocationService’ sends two SOAP requests to extract elements from first item node of the RSS data; one is a date element and another is a title element describing the actual location. For example, to extract the date element, an XPath expression is “//rdf:RDF/item[1]/dc:date”. The extracted values are displayed in the screen as the result of searching the device location.

(3) invokeGimme(ImageViewPad, ‘jpeg’)

Finally, the application sends a SOAP request to get a JPEG image of a Web camera and displays the image on the screen.
Thus, the ‘i-appli’ shows the location as the result of searching from the location service and the photographic image of the Web camera at the location.

5 Discussion

In conventional way, when a user uses both the Device Location Service (DLS) and the Web Camera Service (WCS), the user must switch between two applications which are exclusive for each service. At first, the user obtains the location information by a DLS application and stores the location information in a clipboard for example. After terminating the DLS application to activate a WCS application, the user input the location information to get a photographic image from the WCS.

Those applications are not linked and integrated together so that user operations become complicate and difficult. To integrate separated Web applications into a single linked application, there needs ‘coding’ or ‘scripting’ to make a new application in conventional programming environment. Such program becomes monolithic and inflexible. It is difficult to dynamically change the combination of applications to be integrated.

The IntelligentPad realizes the integration of two or more services, which are served by different servers, into a single linked service dynamically and flexibly as a pad application. The IntelligentPad Web Services makes available such pad application as a single service served at a single server to be used by generic SOAP clients as well as mobile phone applications.

By making the linkage between various services on the server side, client software becomes simple, because the service linkage is executed by composition of pads and the clients should only perform the SOAP communication and screen drawings without data conversion or executing business logic. The ‘i-appli’ cannot communicate with multiple servers by its restriction, but the client of the IPWS can access multiple services indirectly because the linked service is served at a single server. This may reduce development cost of mobile phone applications and expand its application field. Also, service providers may build more sophisticated services by integrating multiple services.
6 Conclusion

We have implemented the IntelligentPad Web Services to expand the application range of our knowledge media technology, in particular to be applied to mobile computing environment. The framework proposed in this paper bridges the IntelligentPad system and other external systems using SOAP. Then, pads enable Web resources to be reusable as well as the pads can be reused by others. During the spread of the new Web technologies, the IntelligentPad can act as a bridge between classical rigid resources and advanced technologies for weaving scattered Web resources. We can expect the further integration of upcoming Web technologies and the IntelligentPad system.

References


