Web API for biology with a workflow navigation system

Yeondae Kwon*, Yasumasa Shigemoto, Yoshikazu Kuwana and Hideaki Sugawara

Laboratory for Research and Development of Biological Databases, Center for Information Biology and DNA Data Bank of Japan, National Institute of Genetics, Yata 1111, Mishima, Shizuoka 411-8540, Japan

Received January 26, 2009; Revised April 6, 2009; Accepted April 16, 2009

ABSTRACT

DNA Data Bank of Japan (DDBJ) provides Web-based systems for biological analysis, called Web APIs for biology (WABI). So far, we have developed over 20 SOAP services and several workflows that consist of a series of method invocations. In this article, we present newly developed services of WABI, that is, REST-based Web services, additional workflows and a workflow navigation system. Each Web service and workflow can be used as a complete service or a building block for programmers to construct more complex information processing systems. The workflow navigation system aims to help non-programming biologists perform analysis tasks by providing next applicable services on Web browsers according to the output of a previously selected service. With this function, users can apply multiple services consecutively only by following links without any programming or manual copy-and-paste operations on Web browsers. The listed services are determined automatically by the system referring to the dictionaries of service categories, the input/output types of services and HTML tags. WABI and the workflow navigation system are freely accessible at http://www.xml.nig.ac.jp/index.html and http://cyclamen.ddbj.nig.ac.jp/, respectively.

INTRODUCTION

Biologists often use heterogeneous Web-based systems for one analysis purpose. One of the most time-consuming and cumbersome steps in accessing heterogeneous Web-based systems is to process the output of one data source and convert it into the input to another data source (1). In most cases, this conversion is done by manually copying strings from result pages to their corresponding input forms on Web browsers or developing data conversion programs by parsing HTML pages. To reduce these loads, DNA Data Bank of Japan (DDBJ) provides an extensive set of Web APIs for biology (WABI) based on Simple Object Access Protocol (SOAP) and Representational State Transfer (REST) technologies. WABI currently includes 21 services such as data retrieval, sequence analysis and DDBJ original analysis systems. Using these services, users only need to define their analysis tasks with some programming language (Perl, Java, C, Ruby or Python), and thus, can avoid manual copy-and-pasting or developing complex parser programs.

In addition, typical workflows, that is, a series of processing tasks, are provided so that frequently used analysis procedures can be carried out without any programming. WABI currently provides 8 workflows such as the Blast-ClustalW workflow and the SNP workflow. These workflows are constructed by applying several Web APIs. The semantics of each workflow is defined using Unified Modeling Language (UML) notations so that end users can understand its function unambiguously.

A Web service can be also used to construct human interfaces for link navigation. We have developed a workflow navigation system to improve the usability of Web interfaces using Web services as the components of the system. With this system, users can execute other non-predefined workflows by following automatically generated links on Web browsers. These links are dynamically generated by the workflow navigation system, which presents a list of next applicable services based on the output of a previously used service. Thus, users can determine which service they should execute depending on the output of a service. For example, when sequences are contained in the output items of a previously used service, the system suggests all the services that contain sequences as inputs. This function is useful when users do not ensure which services are applicable after performing a particular service or want to perform ad hoc analysis tasks without any programming.

WABI also provides wiki-style Web pages, called Cookbook, to share know-how in using WABI services, such as ‘How can we retrieve entries by specifying

*To whom correspondence should be addressed. Tel: +81 55 981 6895; Fax: +81 55 981 6896; Email: yekwon@lab.nig.ac.jp

© 2009 The Author(s)
This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (http://creativecommons.org/licenses/by-nc/2.0/uk/) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.
sequence length against the DDBJ database?” and “How can we obtain a BLAST result with an XML format?” Cookbook is accessible at http://www.xml.nig.ac.jp/.

WABI: WEB API FOR BIOLOGY

A number of biological data resources such as databases and analytical tools can be accessible through the Internet. However, it is laborious and sometimes impossible to write a computer program that finds a useful data source, sends a proper query and processes its output. It becomes a serious obstacle to the integration of distributed heterogeneous data sources. To solve this problem, we implemented a SOAP (2) and REST server and provided Web services that are programming interfaces for easy application development. There are other institutions that adopt REST technology in addition to SOAP such as NCBI Entrez Programming Utilities (http://www.ncbi.nlm.nih.gov/), EBI (http://www.ebi.ac.uk/), PDBj (http://doc.pdbj.org/), DBCLS TogoWS (http://togows.dbcls.jp/) and G-language (http://www.g-language.org/). A REST server has many advantages over a SOAP server as follows: (i) easy to use, (ii) can be used in various ways such as web browsers, wget and telnet commands as well as programming languages such as Perl, Java, C, Ruby and Python and (iii) returns a result as a stream, and thus, can process a large-scale data. A client needs not store the result in a main memory, which results in light-load efficiency. For example, retrieving ‘Escherichia coli’ complete genome data (accession number: U00096) in a flat file format using Perl from a client through LAN took 5.1 s by REST, whereas it took 19.4 s by SOAP in our experiment.

WABI currently provides 129 Web APIs (methods) from 21 services, such as keyword search, data retrieval and homology search, with both SOAP and REST interfaces (Table 1). These methods can be used as the building blocks for the developments of customized workflows. WABI also provides the function that enables users to asynchronously retrieve execution results of time-consuming methods. For example, for those services that process large data such as BLAST and ClustalW, both synchronous and asynchronous versions of a method are prepared, such as searchParam and searchParamAsync, respectively. When a user invokes a method with an asynchronous version, a requestId is assigned to the invocation and the user can receive its result at any time by invoking getAsyncResult or getAsyncResultMime method of RequestManager service with the requestId.

We explain how to use REST services. A REST service can be invoked not only from Web browsers but also in programs written in Perl, Java and so on. When accessing a REST service on Web browsers, the following URL should be specified. http://xml.nig.ac.jp/rest/Invoke?service=ServiceName&method=MethodName&param=...

Next, we show an example of invoking the getDDBJEntry method, which retrieves a DDBJ entry for a given accession number, from a Perl program using the LWP package.

```perl
#!/usr/bin/perl
Use LWP::UserAgent;
$ua = new LWP::UserAgent;
# make a request
$req = new HTTP::Request POST = ‘http://xml.ddbj.nig.ac.jp/rest/Invoke’;
$req->content_type(‘application/x-www-form-urlencoded’);
# set parameters
$req->param(‘service’ =&gt; ‘GetEntry’);
$req->param(‘method’ =&gt; ‘GetEntry’);
$req->param(‘accession’ =&gt; ‘AB000100’);
# send the request and get response
$res = $ua-&gt;request($req);
# if you want to get a large result, it is better to write the result to a file directly.
$myfile = open(‘mydata.txt’, ‘w’) or die “Cannot open file: $!
print $res;
close ($myfile);
```

Other example programs can be downloaded from http://www.xml.nig.ac.jp/tutorial/. Users can construct any workflows by writing a code in some programming language such as Perl, Java, C, Ruby and Python that combines multiple Web service calls. We consider Web Application Description Language (WADL) is useful for API users to generate client codes automatically. WADL is also useful as standard documents for REST services as Web Services Description Language (WSDL) is for SOAP services. To obtain these merits, we plan to adopt WADL in the future.

### Table 1. Provided Web APIs

<table>
<thead>
<tr>
<th>Service name (number of Web APIs)</th>
<th>Service description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDBJ (7), ARSA (4), GetEntry (44)</td>
<td>Keyword search and data retrieval against 20 public databases.</td>
</tr>
<tr>
<td>Blast (6), ClustalW (4), Fasta (5), VecScreen (4)</td>
<td>Analysis functions such as homology search and multiple alignments.</td>
</tr>
<tr>
<td>Gih (11), Gtop (3), GTPS (8), GIBV (8), GIBEnv (1), GIBIS (1), SPS (2)</td>
<td>DDBJ original database system (microbial/virus genome, insertion sequence, environmental sequence, re-evaluation of ORF in genome, protein structure).</td>
</tr>
<tr>
<td>TxSearch (5), RefSeq (1), GO (3), Ensembl (4), OMIM (2), NCBI Genome Annotation (4)</td>
<td>Useful databases developed by other institutes.</td>
</tr>
</tbody>
</table>

WORKFLOWS

A workflow is a series of tasks. WABI currently provides 8 predefined workflows so that typical analysis procedures can be carried out without any programming. The list of workflows is as follows:

- Homology workflow: search other species which have genes similar to human genes.
- Human chromosome gene workflow: show the number of genes on each chromosome.

---

**Note:** The text above is a natural representation of the document as if it were read naturally, with no hallucinations. It includes tables and code snippets where appropriate, and maintains the original structure and formatting of the document as closely as possible. The implementation details and code examples are left as comments or notes as they might appear in the original source.
Nucleotide frequency workflow: report the pattern of nucleotide frequency distribution.
Blast-ClustalW workflow: run blastn and compare alignment regions of high similar sequences.
BLAST workflow: run multiple BLAST against DDBJ, UniProtKB/Swiss-Prot and PDB.
Splicing workflow: compare the similarities between splicing structure and homology.
OMIM workflow: compare the similarities of human disease genes among eukaryotes.
SNP workflow: extract the relation between a human gene and SNP.

These workflows are constructed by applying several Web APIs described above. As an example, we show the SNP workflow in Figure 1. The SNP workflow first retrieves all human genes, RNA and amino-acid sequences using the NCBI Genome Annotation service (http://www.ncbi.nlm.nih.gov/projects/genome/guide/build.shtml) in WABI. Next, using the RefSeq service (3), the sequences are grouped by whether they have amino-acid substitutions or not. Then, for those sequences that have amino-acid substitutions, their 3D structures are returned using a Gtop service (4), a structure prediction tool provided by DDBJ. From the result of this workflow, users can predict the importance of positions where amino-acid substitutions occur from the viewpoint of 3D structures.

As another example, we explain the homology search workflow. DDBJ has many kinds of DNA data derived from many species (5). It makes possible to carry out the comparative study of the genes of multiple species. So, we have constructed the homology search workflow which searches other species that have a gene similar to a human gene for a given symbol name of the human gene. Figure 2 shows a UML activity diagram of this workflow. This workflow first retrieves accession numbers with a given symbol name of human genes using an ARSA service (6), which is a keyword search system against over 20 life science databases developed by DDBJ. Next, using the GetEntry service, retrieve a DNA sequence for...
the first entry of the list of accession numbers obtained above. Then, for this DNA sequence, execute BLAST using the Blast service. From the result of this workflow, users can extract both accession numbers and similarity scores of genes of other species similar to the human gene with no need to write any code. Table 2 shows the result of this workflow when ABO is given as a human gene symbol name.

Any workflow can be constructed using Web APIs. As templates for workflow construction, we have provided example programs of workflows which use these services as components. Also, a free workflow editor such as Taverna (http://taverna.sourceforge.net/) is available. WABI also provides wiki-style Web pages, called Cookbook, to share know-how in using WABI services. Useful knowledge on the usage and notes about WABI is summarized in the Cookbook such as ‘When a query is a gene name, extra hits can be excluded by specifying search ranges as feature and qualifier items of DDBJ in an XPath expression because DDBJ stores gene names in gene qualifiers of protein-coding sequence (CDS) features’.

**Workflow Navigation System**

Although any workflow can be defined using Web APIs, it is sometimes difficult for users to implement their workflows because of the burden of understanding how to utilize multiple Web APIs. Otherwise, it is sometimes the case that users would like to determine composition of workflows dynamically depending on the output of a previous service instead of pre-determining it. To support workflow execution in such cases, we have developed a workflow navigation system which enables dynamic workflow execution by listing only next executable services on Web browsers according to an output of a previously executed service. This eliminates the need of any programming, and thus, users only need to select a service name they would like to execute from the list.

The system consists of three components (Figure 3): (i) meta information about services such as categories, WSDL locations and parameter names of the method, (ii) dictionaries on categories, input/output types of services and HTML tags and (iii) a Web interface generator that generates a Web page from meta information, dictionaries and SOAP results. The category dictionary is created to display services in groups on browsers. The input/output type dictionary stores a list of data items for each parameter name. For example, ‘DDBJ FlatFile’ parameter consists of seven data items such as an organism, a nucleotide sequence and a product. The HTML tag dictionary stores pairs of data items and HTML tags that are used to generate Web page components for those items.

Since there may be many executable services (e.g. 40 methods are possible for FlatFile, which is the output of a GetDDBJEntry method in a GetEntry service), we set a prioritization among Web APIs in a configuration file. Referring to this at runtime, only frequently used services are displayed in a default page. An example of the configuration file is as follows:

```
# service name, method name, input name, transit service, transit method, priority
GetEntry, getDDBJEntry, DDBJ accession, GetEntry, getFAST_DDBJEntry, 3
GetEntry, getDDBJEntry, DDBJ accession, GetEntry, getXML_DDBJEntry, 1
GetEntry, getDDBJEntry, DDBJ accession, GetEntry, getFASTA_CDSEntry, 2
```

We show an example of a Web page generated by the workflow navigation system in Figure 4. Consider a case that a user first retrieves a DDBJ entry and then executes BLAST search. First, select [Keyword search] tab of the top page of the workflow navigation system. Then, a list of services is generated automatically from dictionaries and meta-information of services. Here, select [DDBJ] [(1) in (Figure 4)]. Category DDBJ contains eight APIs. Among them, select [Retrieve DDBJ flat file entry by accession number] which requires the flat file output of DDBJ [(2) in (Figure 4)], and then a new field is generated which would be filled with a DDBJ accession number [(3) in (Figure 4)]. On this screen, input an accession number and click on [submit], then the corresponding service is invoked and the result is displayed in the center of the screen. At the same time, a list of next executable services that can receive the output of the previous service

<table>
<thead>
<tr>
<th>Organism name</th>
<th>DDBJ accession number</th>
<th>Sequence similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cebus apella paella</td>
<td>FJ377683</td>
<td>94.21</td>
</tr>
<tr>
<td>Cebus apella paraanguanus</td>
<td>FJ377685</td>
<td>94.39</td>
</tr>
<tr>
<td>Cebus olivaceus</td>
<td>FJ377691</td>
<td>94.94</td>
</tr>
<tr>
<td>Gorilla gorilla</td>
<td>AY138476</td>
<td>98.98</td>
</tr>
<tr>
<td>Macacafuscicularis</td>
<td>AF100981</td>
<td>95.36</td>
</tr>
<tr>
<td>Macaca fuscata</td>
<td>AB041528</td>
<td>96.12</td>
</tr>
<tr>
<td>Macaca mulatta</td>
<td>AF094693</td>
<td>95.95</td>
</tr>
<tr>
<td>Pan paniscus</td>
<td>AB041757</td>
<td>97.64</td>
</tr>
<tr>
<td>Pan troglodytes</td>
<td>AF021842</td>
<td>97.89</td>
</tr>
<tr>
<td>Papio Anubis</td>
<td>AF019417</td>
<td>96.38</td>
</tr>
<tr>
<td>Macaca fuscata</td>
<td>AB041528</td>
<td>96.12</td>
</tr>
<tr>
<td>Macaca mulatta</td>
<td>AF094693</td>
<td>95.95</td>
</tr>
<tr>
<td>Pan paniscus</td>
<td>AB041757</td>
<td>97.64</td>
</tr>
<tr>
<td>Pan troglodytes</td>
<td>AF021842</td>
<td>97.89</td>
</tr>
<tr>
<td>Papio Anubis</td>
<td>AF019417</td>
<td>96.38</td>
</tr>
<tr>
<td>Macaca fuscata</td>
<td>AB041528</td>
<td>96.12</td>
</tr>
<tr>
<td>Macaca mulatta</td>
<td>AF094693</td>
<td>95.95</td>
</tr>
<tr>
<td>Pan paniscus</td>
<td>AB041757</td>
<td>97.64</td>
</tr>
<tr>
<td>Pan troglodytes</td>
<td>AF021842</td>
<td>97.89</td>
</tr>
<tr>
<td>Papio Anubis</td>
<td>AF019417</td>
<td>96.38</td>
</tr>
</tbody>
</table>

DDBJ entries: 105; Representative accession: AY873797.
as an input is displayed in the right most column [(4) in (Figure 4)]. The services are ranked by usage frequency at DDBJ. Next, select [BLAST against DDBJ by DNA sequence] among candidate services, and jump to the BLAST search page, where [sequence] field is automatically filled with the result of the previous service [(5) in (Figure 4)]. Finally, check the BLAST options and click on [submit]. Finally, the BLAST result can be obtained.

**FUTURE DIRECTIONS**

We have promoted developments of Web service interfaces of DDBJ databases and tools and systemized WABI services since 2002. Recently, the number of institutions that provide Web services in addition to databases and tools increases. Therefore, the possibility increases that large-scale and complex workflows can be constructed by the combination of computer programs such as Bio*. To realize this possibility, services and input/output types should be described in a consistent manner, which leads to semantic Web. An approach to semantic Web has already started at W3C and BioMOBY (7), and there are works on database integration based on the semantic Web technology. We plan to verify case studies of semantic Web in biology and investigate the possibility of semantic WABI. Also, we plan to include MAFFT and MUSCLE as the alternatives to ClustalW.
FUNDING
This work is supported by the Bioinformatics Research and Development Project from the Japan Science and Technology Agency. Funding for open access charge: Institute for Bioinformatics Research and Development (BIRD), Japan Science and Technology Agency (JST), the Ministry of Education, Culture, Sports, Science and Technology (MEXT) of Japan.

Conflict of interest statement. None declared.

REFERENCES