Online journal ISSN - 1676-5680
GMR Genetics and Molecular Research
EVOLUTION AND TECHNOLOGY

Development of SRS.php, a Simple Object Access Protocol-based library for data acquisition from integrated biological databases

A. Barbosa-Silva[1,2], E. Pafilis[2], J.M. Ortega[1] and R. Schneider[2]

[1]Departamento de Bioquímica e Imunologia,
Universidade Federal de Minas Gerais, Belo Horizonte, MG, Brasil
[2]Structural and Computational Biology Unit,
European Molecular Biology Laboratory, Heidelberg, Germany
Corresponding author: J.M. Ortega
E-mail: miguel@icb.ufmg.br

Genet. Mol. Res. 6 (4): 1142-1150 (2007)
Received August 30, 2007
Accepted November 20, 2007
Published December 11, 2007

ABSTRACT. Data integration has become an important task for biological database providers. The current model for data exchange among different sources simplifies the manner that distinct information is accessed by users. The evolution of data representation from HTML to XML enabled programs, instead of humans, to interact with biological databases. We present here SRS.php, a PHP library that can interact with the data integration Sequence Retrieval System (SRS). The library has been written using SOAP definitions, and permits the programmatic communication through webservices with the SRS. The interactions are possible by invoking the methods described in WSDL by exchanging XML messages. The current functions available in the library have been built to access specific data stored in any of the 90 different databases (such as UNIPROT, KEGG and GO) using the same query syntax format. The inclusion of the described functions in the source of scripts written in PHP enables them as webservice clients to the SRS server. The functions permit one to query the whole content of any SRS database, to list specific records in these databases, to get specific fields from the records, and to link any record among any pair of linked databases. The case study presented exemplifies the library usage to retrieve information regarding registries of a Plant Defense Mechanisms database. The Plant Defense Mechanisms database is currently being developed, and the proposal of SRS.php library usage is to enable the data acquisition for the further warehousing tasks related to its setup and maintenance.

Key words: Data integration, Web services, XML, SOAP, Sequence Retrieval System

INTRODUCTION

The life sciences community has experienced an enormous explosion of data generation in the last 20 years. Several of the existing biological databases (bioDBs) in the world usually double their content in periods of less than 2 years. Besides the generation of new datasets, there are issues related to i) the ambiguity of information provided by distinct collections, and ii) the manner that distinct data is shared among the data providers. Such problems are currently addressed by a field in bioinformatics known as data integration and management.

The strategies used to share the content of the bioDBs from the servers to the users usually fit in two categories: those directed to interact with humans users, through user-friendly interfaces, gathering query inputs using simple HyperText Markup Language (HTML) forms and answering them by generating dynamic web pages, and those created to interact specifically with programs, i.e., to scripts, using eXtensible Markup Language (XML). The basic difference between the two languages is that in using XML, the elements contained in the information to be displayed are referenced by tags that determine each piece of information, while in using just HTML one is not able to extract the semantic meaning of a term in the documents (Achard et al., 2001). If the code for some database record is written in XML, the data would receive both: format information (e.g., the color green in the browser) and contextual information (e.g., if the displayed data are a protein or gene name).

The use of XML files turn the database records adapted to receive queries made by programs instead of humans (Wang et al., 2002). On the other hand, because it is an encoding language as clear as its predecessor HTML, it maintains the human readability as well.

Currently, important questions that arise in biology are no longer possible to be answered by just one database (Stein, 2003). Instead, researchers tend to jump from one database to another to collect if the information that would help shed light on the formulated problem. This is the normal way the access is done manually; however, using XML in the bioDBs, an environment is created that can remarkably reduce the need of ‘database surfing’. The effort of getting split information is transferred to scripts that directly interact with the servers and retrieve the information in a much more efficient way, in this case using webservices technology.

The workflows commonly used in biology fit well in the data exchange schema proposed by the webservices (Curcin et al., 2005). This is an interface that allows programmatic access, i.e., application to application, to the data stored in the bioDBs (Romano et al., 2005). Webservices are dependent on some XML-based standards such as: Simple Object Access Protocol (SOAP), a protocol used to exchange structured information in a decentralized network; the Web Services Description Language (WSDL), an XML format which the data provider uses to describe the services available in the server, and a Universal Description Discovering and Integration (UDDI) registry, directed for providers to publish their services and for users to locate the desired webservices available on the Internet.

Currently, diverse biological resources are using the webservices model to offer their tool or release their stored data for users (e-Utilities; Miyazaki et al., 2004; NCICB). One of these resources is the EMBL-SRS Universe (http://srs.embl.de) (Zdobnov et al., 2002). It is possible to query and navigate through several databases available in the server by using the Sequence Retrieval System (SRS) of this machine. Additionally, the users can exploit the link between the databases to precisely extract the information from multiple sources. The aim of the present study was to describe SRS.php, a computational resource designed to integrate dynamic scripts written in Hypertext Preprocessor (PHP) to the SRS available by the EMBL-Heidelberg server. This is done by using SOAP-based webservice requests that communicate directly with the system by changing messages based in XML. This resource is released as an open source library: a set of functions that can be embedded in the source code of scripts written to perform specific tasks. The utility of the library is illustrated by the usage of SRS.php in the acquisition of specific information from multiple sources for sequences deposited in a bioDB developed by us, the Plant Defense Mechanisms Database.

MATERIAL AND METHODS

Features of target Sequence Retrieval System

The aim of the library proposed in this study was to interact with the SRS made available by the EMBL-Heidelberg; the server is composed of approximately 90 different biological databases (http://srs.embl.de/srs/databanklist.do). The databases have a description webpage where precise information about the features of each of them can be accessed. The information contains the database description, the field list for each record, the database loaders, and the target databases that the selected database links in the same system.

Record features

The records deposited in a database under an SRS have a peculiar syntax that deserves comment before the explanation of how it is accessed through SOAP. Each item is deposited as a table in which the rows are the attributes of such record. The fields are abbreviated by a short name that facilitates the database querying. An example of a typical record of the database UNIPROT (Apweiler et al., 2004) is illustrated briefly in Table 1.

Table 1. Brief representation of the UNIPROT database installed in a Sequence Retrieval System.

Database name UNIPROT		
Field name	Short name	Content
Identifier	id	Q9XET3_SOLLC
Prim. Accession	pac	Q9XET3
Description	des	Disease resistance protein I2
Organism	org	*Solanum lycopersicum* (Tomato) (*Lycopersicon esculentum*)
NCBI Taxonomy ID	txi	4081

Query syntax

One who wants to get the information of a record deposited in UNIPROT database from an SRS running on a Linux platform, for example, should do it from the command line using the program getz (see SRS documentation available at http://srs.embl.de/srs/doc/srsbooks/srs_user_guide/21_1.html). This program receives several parameters, but in order to describe the usage of our library, we will focus on just some of them.

Listing records from some database

To display all the records deposited for the organism exemplified in Table 1, one should submit the getz query in Formula 1, this would retrieve all the records belonging to the organism identified by the NCBI Taxonomy id (txi) 4081 in the UNIPROT database.

```
[srs_server]$ getz "[UNIPROT-txi:4801]"
```
(Formula 1)

Displaying records from some database

In case the user wants to display a specific record from some database, it is necessary to use some optional flags in the getz program. The following command uses the flag '-e' in order to invoke the loader CompleteEntry which accesses and displays the whole record identified by the accession number Q9XET3.

```
[srs_server]$ getz "[UNIPROT-acc:Q9XET3]" –e
```
(Formula 2)

Access specific fields

Another command flag of the getz program allows users to retrieve from the command line the specific fields of some record. In Formula 3, the operator '-vf' is used for this purpose. In the example, the Medline database is queried and the field 'authors' (aut) is chosen to be displayed for the article identified by the Medline ID (id) 8208723.

```
[srs_server]$ getz "[medline-id:8208723]" –vf  aut
```
(Formula 3)

Exploring the link universe

The SRS feature used to link the records among the distinct databases can be explored using the getz program as well. For this, the operator ">" is used in Formula 4; with the illustrated command, the article from the Medline database identified by the ID 15371431 is linked to its corresponding record in the UNIPROT database.

```
[srs_server]$ getz "[medline-id: 15371431]>UNIPROT"
```
(Formula 4)

Implementation of SRS.php library

The SRS.php library was written in PHP 5 and has been tested in a system running the same or superior version. However, the command line interface has been implemented in PHP since version 4.3.0, for this reason it is expected that the library should also work with versions as old as this. In some machines, it is possible that some warnings or error messages could be displayed on the screen; to avoid this, users must set the *error_reporting* variable in the php.ini configuration file for “E_ALL & ~E_NOTICE | E_STRICT” for instance. Another feature that can affect the library performance is the resource limits of the PHP installed in the machine; in order to avoid problems in regard to this issue, the variables related to these limits should be set to compatible values.

The library is expected to enable scripts that include it on the code to act as SOAP clients to the EMBL-SRS server. That is why it uses the SOAP features available through the class nuSOAP, which can be freely downloaded from the developers’ webpage (nuSOAP).

RESULTS AND DISCUSSION

Description of the library

The initial core of request methods present in the SRS.php library consists of 4 functions that allow the data integration between the SRS database engine and PHP dynamic scripts. These functions interact with SRS based on information found on its WSDL description file. This file is accessed using the *soapclient* nuSOAP function, which receives the WSDL file location as parameter (http://srs8.embl.de:8989/axis/services/SrsWrapper?wsdl); this is the common feature present in all functions which are described below.

Acquisition of number of results for queries

To access the number of results for one request for any of the SRS databases, we created the function *performIcarusQueryAndGetNumberOfResults* (Figure 1). This function receives as parameter one string corresponding to the getz query in Formula 1, and returns a one-dimensional array with the number of records that matched the request in SRS (Figure 5A, 1). As an example, one can obtain the present number of entries in Uniprot for a given organism.

```
function performIcarusQueryAndGetNumberOfResults($query){

  // Use the server URL
  global $server;

  // Create instance to server
  $client = new soapclient($server);

  // Fill the parameters for the service
  $param=array("query"=>$query);

  // Submit the query to the web-service
  $results = $client -> call("performIcarusQueryAndGetNumberOfResults",$param);

  return($results);
}

#EXAMPLE
#$num=performIcarusQueryAndGetNumberOfResults("[UNIPROT-txi:3702]");
#echo $num[0];
```

Figure 1. Function performIcarusQueryAndGetNumberOfResults.

Accessing records using loaders

The second function, *performQueryAndGetLoaderDefinedFields*, was created to allow the access to the full records deposited in the SRS databases, either directly or using the Link Universe facility as well. As parameter, one must supply the getz query (like in Formula 2) and the name of the loader, a tool to extract specific information from each SRS entry, to be used (Figure 2). The function returns a one-dimensional array with the content of the loader result (i.e., defined attributes for one SRS entry) as output (Figure 5A, 2). For instance, it is possible from a UNIPROT accession number to retrieve the complete entry of a record using the loader CompleteEntry.

```
function performQueryAndGetLoaderDefinedFields($query,$loader){

  // Use the server URL
  global $server;

  // Create instance to server
  $client=new soapclient($server);

  // Fill the parameters for the service
  $param=array("query"=>$query,
                "loader"=>$loader,
                );

  // Submit the query to the web-service
  $results = $client -> call("performQueryAndGetLoaderDefinedFields",$param);

  // Result from the function
  return($results);
}
#EXAMPLE
#$result=performQueryAndGetLoaderDefinedFields("[UNIPROT-acc:Q9XET3]","CompleteEntry");
#echo $result[0];
```

Figure 2. Function performQueryAndGetLoaderDefinedFields.

Accessing specific fields in the records

The content of specific attributes for one SRS record can be accessed using the third function, *performQueryAndGetSpecificFields* (Figure 3). This function needs four parameters: i) the target database to be queried, ii) the target field, iii) the query term, and iv) the fields to be displayed from the accessed record. The function returns the specific fields that matched the term (Figure 5A, 3). The operation performed by this function is equivalent to the getz query of Formula 3.

```
function performQueryAndGetSpecificFields($queryTargetDb,$queryTargetField,$queryTerm,$fields){

  // Use the server URL
   global $server;

  // Create instance to server
  $client=new soapclient($server);

  // Fill the parameters for the service
  $param=array("queryTargetDb"=>$queryTargetDb,
                "queryTargetField"=>$queryTargetField,
                "queryTerm"=>$queryTerm,
                "fields"=>$fields
                );

  // Submit the query to the web-service
  $results = $client -> call("performQueryAndGetSpecificFields",$param);

  // Removes the inicial field of the getz query
  $results[0]=preg_replace("/\s+/","#",$results[0]);
  $results=explode("#",$results[0]);
  array_shift($results);
  $field=implode(" ",$results);

  // Result from the function
  return($field);
}

#EXAMPLE
#$field=performQueryAndGetSpecificFields("medline","id","8208723","aut");
#echo $field;
```

Figure 3. Function performQueryAndGetSpecificFields.

Exploring the Sequence Retrieval System link universe

This special characteristic of the SRS was included in the library by the fourth function, *performLinkingQuery* (Figure 4). This function, like the query in Formula 4, receives as parameters the database in which the query term is deposited, the parameter (field) to which the query refers, the query term and the database to which the query should be linked. The function above returns the record IDs in the linked database related to the term provided by the function (Figure 5A, 4). For example, all the INTERPRO domain signatures of a UNIPROT sequence could be retrieved by using this function, through the SRS Link Universe resource.

```
function performLinkingQuery($queryTargetDb,$queryTargetField,$queryTerm,$linkingTargetDb){

  // Use the server URL
  global $server;

  // Create instance to server
  $client=new soapclient($server);

  // Fill the parameters for the service
  $param=array("queryTargetDb"=>$queryTargetDb,
               "queryTargetField"=>$queryTargetField,
               "queryTerm"=>$queryTerm,
               "linkingTargetDb"=>$linkingTargetDb
               );

  // Submit the query to the web-service
  $results = $client -> call("performLinkingQuery",$param);

  // Result from the function
  return($results);
}

#EXAMPLE
#$link=performLinkingQuery("medline","id","15371431","UNIPROT");
#for($i=0;$i<sizeof($link);$i++) print "$link[$i].<BR>";
```

Figure 4. Function performLinkingQuery.

Figure 5A summarizes the functions present in the SRS.php library. A case study using the functions mentioned above is reported in the next session.

Case study: data warehousing for Plant Defense Mechanisms database

One of us (A.B.-S.) is developing a database which collects proteins that are involved in Plant Defense Mechanisms (PDM). Currently, the database contains sequences from UNIPROT database (seed sequences), and collected similarity-related sequences (putative orthologs) to the seed sequences through a Seed Linkage clustering approach (Barbosa-Silva A, Satagopam VP, Schneider R and Ortega JM, unpublished results) based on bidirectional best-hit strategy.

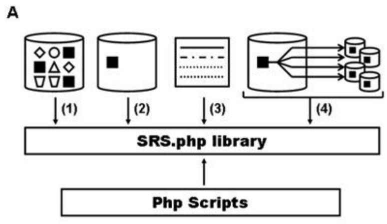
To improve the annotation of the data deposited in the PDM database, we have integrated the information from diverse set of databases.

Using the SRS.php library, we first queried for each plant represented in the PDM database, about its total number of entries deposited in the following databases: UNIPROT, RefSeq, UniRef (100, 90 and 50), and PIR (Figure 5B).

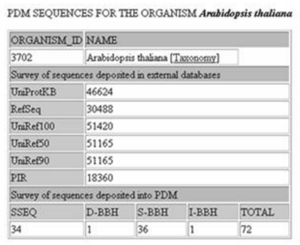
In a second step, we used the SRS.php library to access the annotations in the UNIPROT, GO, PFamA, INTERPRO, and Prosite databases, which can be related to the PDM sequences using their UNIPROT ID as query (Figure 5C). Interested users can download the script used for these tasks, retrieve_srs.php, from the supplementary material (http://www.biodados.icb.ufmg.br/seed_linkage).

The usage of SRS.php functions in the annotation of PDM database sequences was a very straightforward approach, since the acquisition of further information on demand for the PDM sequences was very fast, compared to manual annotation, considering the existence of only the UNIPROT ID as starting point. Hitherto, we have added several forms of useful information that increased undoubtedly the content and quality of the PDM entries.

A



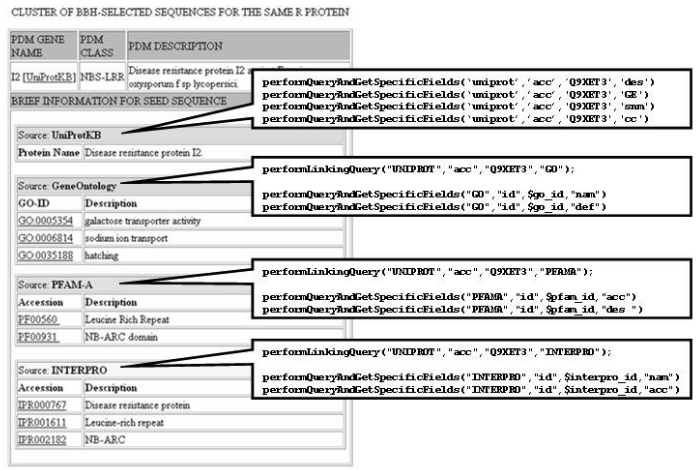
B



PDM SEQUENCES FOR THE ORGANISM *Arabidopsis thaliana*

ORGANISM_ID	NAME			
3702	Arabidopsis thaliana [Taxonomy]			
Survey of sequences deposited in external databases				
UniProtKB	46624			
RefSeq	30488			
UniRef100	51420			
UniRef50	51165			
UniRef90	51165			
PIR	18360			
Survey of sequences deposited into PDM				
SSEQ	D-BBH	S-BBH	I-BBH	TOTAL
34	1	36	1	72

C



CLUSTER OF BBH-SELECTED SEQUENCES FOR THE SAME R PROTEIN

PDM GENE NAME	PDM CLASS	PDM DESCRIPTION
I2 [UniProtKB]	NBS-LRR	Disease resistance protein I2 [illegible] oxysporum f sp lycopersici

BRIEF INFORMATION FOR SEED SEQUENCE

Source: UniProtKB	
Protein Name	Disease resistance protein I2.

```
performQueryAndGetSpecificFields('uniprot','acc','Q9XET3','des')
performQueryAndGetSpecificFields('uniprot','acc','Q9XET3','GE')
performQueryAndGetSpecificFields('uniprot','acc','Q9XET3','snm')
performQueryAndGetSpecificFields('uniprot','acc','Q9XET3','cc')
```

Source: GeneOntology

GO-ID	Description
GO:0005354	galactose transporter activity
GO:0006814	sodium ion transport
GO:0035188	hatching

```
performLinkingQuery("UNIPROT","acc","Q9XET3","GO");

performQueryAndGetSpecificFields("GO","id",$go_id,"nam")
performQueryAndGetSpecificFields("GO","id",$go_id,"def")
```

Source: PFAM-A

Accession	Description
PF00560	Leucine Rich Repeat
PF00931	NB-ARC domain

```
performLinkingQuery("UNIPROT","acc","Q9XET3","PFAMA");

performQueryAndGetSpecificFields("PFAMA","id",$pfam_id,"acc")
performQueryAndGetSpecificFields("PFAMA","id",$pfam_id,"des ")
```

Source: INTERPRO

Accession	Description
IPR000767	Disease resistance protein
IPR001611	Leucine-rich repeat
IPR002182	NB-ARC

```
performLinkingQuery("UNIPROT","acc","Q9XET3","INTERPRO");

performQueryAndGetSpecificFields("INTERPRO","id",$interpro_id,"nam")
performQueryAndGetSpecificFields("INTERPRO","id",$interpro_id,"acc")
```

Figure 5. A. Using the library SRS.php, the user can (1) count the number of matches (small black squares) in one Sequence Retrieval System (SRS) database (DB1), (2) retrieve specific records (black square), and (3) retrieve specific attributes of each deposited record (different lines). Additionally (4), the user can link the record (small black square) to the universe of SRS databases. **B.** Number of records deposited for *Arabidopsis thaliana* in different databases. **C.** Retrieved annotations for gene I2 (UNIPROT accession: Q9XET3) of *Lycopersicon esculentum* in Plant Defense Mechanisms (PDM) from different databases (UNIPROT, GO, PFam-A, INTERPRO); the screenshots represent examples of the SRS.php information retrieval for PDM records.

CONCLUSIONS

Integration of biological databases is a current issue addressed by several data providers in life sciences (Stein, 2002). Currently, webservices is the default technology chosen for data integration not only in biology, but also for other internet resources when the information is not centered on just one source. A unified language that permits biological information to be shared in this scenario is XML (Achard et al., 2001). Hitherto, an increasing number of open source applications have been developed to explore the facilities of webservices. We provide SRS.php as an open source library that gives support for the development of new dynamic scripts that can act as true data retrievers through a programmatic access to several bioDBs under the SRS hosted by EMBL. In a study case, information integration provided automated maintenance of Protein Defense Mechanism Database with information contained in several bioDBs such as UNIPROT, INTERPRO, PFam, etc. In the future, following the development of new services provided by SOAP interface of SRS, plus the feedback received from users, we aim to improve the content and functionalities of SRS.php library.

AKNOWLEDGMENTS

The authors are thankful to Theodorus Soldatos, Georgios Pavlopoulos and Venkata Satagopam for providing training at EMBL to A. Barbosa-Silva and CAPES for providing the PDEE fellowship to develop this research in association with EMBL; to EMBL for providing scientific resources that enabled the execution of this study, and to FAPEMIG for supporting our research.

REFERENCES

Achard F, Vaysseix G and Barillot E (2001). XML, bioinformatics and data integration. *Bioinformatics* 17: 115-125.

Apweiler R, Bairoch A, Wu CH, Barker WC, et al. (2004). UniProt: the Universal Protein knowledgebase. *Nucleic Acids Res.* 32: D115-D119.

Curcin V, Ghanem M and Guo Y (2005). Web services in the life sciences. *Drug Discov. Today* 10: 865-871.

e-Utilities - Entrez Utilities Web Service. http://www.ncbi.nlm.nih.gov/entrez/query/static/esoap_help.html. Accessed August 23, 2007.

HTML. HyperText Markup Language. http://www.w3.org/TR/html401/. Accessed August 23, 2007.

Miyazaki S, Sugawara H, Ikeo K and Gojobori T (2004). DDBJ in the stream of various biological data. *Nucleic Acids Res.* 32: D31-34.

NCICB. National Cancer Institute Web Services. http://ncicb.nci.nih.gov/infrastructure/cacore_overview. Accessed August 23, 2007.

nuSOAP. http://dietrich.ganx4.com/nusoap/. Accessed August 23, 2007.

PHP. Hypertext Preprocessor. http://www.php.net. Accessed August 23, 2007.

Romano P, Marra D and Milanesi L (2005). Web services and workflow management for biological resources. *BMC Bioinformatics* 6 (Suppl 4): S24.

SOAP. Simple Object Access Protocol Specifications. http://www.w3.org/TR/soap/. Accessed August 23, 2007.

Stein L (2002). Creating a bioinformatics nation. *Nature* 417: 119-120.

Stein LD (2003). Integrating biological databases. *Nat. Rev. Genet.* 4: 337-345.

UDDI. Universal Description, Discovery and Integration. http://www.uddi.org. Accessed August 23, 2007.

Wang L, Riethoven JJ and Robinson A (2002). XEMBL: distributing EMBL data in XML format. *Bioinformatics* 18: 1147-1148.

WSDL. Web Services Description Language. http://www.w3.org/TR/wsdl. Accessed August 23, 2007.

XML. Extensible Markup Language. http://www.w3.org/XML/. Accessed August 23, 2007.

Zdobnov EM, Lopez R, Apweiler R and Etzold T (2002). The EBI SRS server - new features. *Bioinformatics* 18: 1149-1150.