2010 10th IEEE International Conference on Advanced Learning Technologies

A proposal to improve the Simple Query Interface (SQI) of Learning Objects repositories

Salvador Otón, José R. Hilera, Eva García, Antonio García, Luis de-Marcos, Antonio Ortíz, José A. Gutiérrez, José J. Martínez, José M. Gutiérrez, Roberto Barchino
Computer Science Department
University of Alcalá
Alcalá de Henares, Madrid, Spain
salvador.oton@uah.es

***Abstract*—This paper proposes to improve the Simple Query Interface (SQI), designed to query learning object repositories, modifying a few existing methods to generate new ones. Replacement is not intended, instead, our proposal is to create new methods that incorporate several enhancements in order to enable compatibility with currently compliant systems that use the old methods. Besides, new methods are also proposed aiming at improving the interface as well as at providing compatibility with the Simple Publishing Querying (SPI) specification. With these improvements SQI specification will perform its task in a more structured and efficient manner.**

Keywords-SQI SPI, interface, e-learning, repositories

I. INTRODUCTION

Simple Query Interface (SQI) was published by the European Committee for Standarization in 2005. Its objective is to facilitate interoperability between public learning object repositories and the applications that use them [1, 2].

SQI interface comprises the set of methods provided by a repository, in such a way that remote clients may query over the learning objects stored in such a repository. These methods enable to perform different operations, but it is possible to improve a few of them.

Specifically, a modification of the methods *synchronousQuery* and *asynchronousQuery* is proposed without removing them, so that compatibility with older systems is kept. Besides, we propose to add two new methods, namely *downloadResource* and *asynchronousDownloadResource*, that enable direct download of learning objects synchronously or asynchronously.

The state of the art and some comparative evaluation results are presented in [3].

Section II presents an introduction to SQI interface core concepts. Section III introduces the first proposal for upgrading (modification existing methods). Section IV presents the second proposed improvement (addition of the *downloadResource* method along with its asynchronous counterpart). Finally Section V provides conclusions.

II. SIMPLE QUERY INTERFACE

A. SQI methods

SQI specification comprises thirteen methods divided into three groups: configuration methods, session management methods and query methods. Query methods can be further divided into synchronous query methods and asynchronous query methods.

Table I presents new SQI methods proposed to improve the specification.

TABLE I. CLASSIFICATION OF SQI METHODS, INCLUDING THE NEW METHODS PROPOSED

QUERY METHODS	
SYNCHRONOUS QUERY	***ASYNCHRONOUS QUERY***
synchronousQueryObject (sessionID, queryStatement, startResult): queryResults	asynchronousQueryObject (sessionID, queryStatement, queryID)
	queryResultsListenerObject (queryID, queryResults)
downloadResource (sessionID, resourceID, downloadType): byte[]	asynchronousDownloadResource (sessionID, queryStatement, queryID)
	downloadResourceResultsListener (queryID, queryResults)

B. SQI fault codes

SQI interface also describes seventeen fault codes that current methods may throw. Three new fault codes have been incorporated because they are used by the new methods. Table II presents new fault codes.

TABLE II. NEW SQI FAULT CODES PROPOSED

CODE	DESCRIPTION
SQI-00017	Invalid resource identifier
SQI-00018	Download IO error
SQI-00019	Download mode not supported

It worth mentioning that fault code SQI-00017 corresponds to fault code SPI-00005 of the SPI specification. SPI is used to publish contents in repositories [4]. So our

978-0-7695-4055-9/10 $26.00 © 2010 IEEE
DOI 10.1109/ICALT.2010.25

proposal also provides compatibility and homogeneity with the SPI specification.

III. MODIFICATION SYNCHRONOUSQUERY, ASYNCHRONOUSQUERY AND QUERYRESULTSLISTENER METHODS

The first proposal deals with *synchronousQuery* and *asynchronousQuery* existing methods.

A. Modification synchronoysQuery method

Existing *synchronousQuery* method returns a String, which is not the most appropiate to return metadata of a set of learning objects that match with search parameters inserted by user. Each learning object is specified by the tags that match with the standard of meta-information used (in this case, LOM), and it includes the address where it is located.

Obviously, a String is not the optimum way to manage large amounts of information, because the volume of learning objects that a repository can return as response to a query can be very important. That makes rather laborious its management, filtering and analysis. To facilitate the task of filtering, processing and ordering learning objects, we propose to create an object that contains at least the fourteen top used fields [5] and to return an object list instead of a string.

Using this system provides several advantages to analyze the returned information. Some of them are:

- Ordering their values by any of fields is a trivial task, because it is a list of linked data.
- Removing information is also an easy and quick task.
- If binary trees were used, this would improve search systems within large quantities of learning objects.

Modification of *synchronousQuery* method, therefore, results in the creation of a new method, named *synchronousQueryObject*, whose header is shown in fig. 1.

Method name	**synchronousQueryObject**	
Return type	List<LOM>	
Parameters	*Name*	*Type*
	targetSessionID	String
	queryStatement	String
	startResult	Integer
Fault	NO_SUCH_SESSION INVALID_ QUERY_STATEMENT QUERY_MODE_NOT_SUPPORTED METHOD_FAILURE INVALID_START_RESULT NO_MORE_RESULTS METHOD_FAILURE	

Figure 1. Header of synchronousQueryObject method proposed

Input parameters match with those of *synchronousQuery* existing method, just the return type changes. Therefore, parameter *queryResults* of *synchronousQueryObject* method of table I will be a list of objects that comply with chosen standard (in this case, LOM), instead of a String, as in the case of *synchronousQuery*.

As mentioned above, modification of method does not pretend to replace the original, but to create a new method with a different name, to keep the compatibility with systems that are using the old method.

B. Modification asynchronousQuery method

To make a query comply with this improvement (that is, obtaining lists of LOM objects instead of strings as result of query), it is necessary to know how asynchronous queries work: the first thing to do is call the *setSourceLocation* method to locate the resource, later an asynchronous query is launched with the *asynchronousQuery* method and, finally, the client executes *queryResultsListener* method in order to get the query result. The latter is the only method that is executed on the client, the others are executed on the server.

The *setSourceLocation* method can be reused for the improvement proposed here, because it only locates where the wanted resource is placed.

However, it is necessary to modify the other two methods to create other two new methods that match with them, but that include the improvement proposed. As mentioned above, modification will not imply removing old methods, but new ones will be created to maintain the compatibility of systems that still use old methods.

1) asynchronousQueryObject

It is necessary to add of a new asynchronous method that matches with *asynchronousQuery* existing method but whose queries finally get lists of LOM objects instead of strings. This new method is named *asynchronousQueryObject*, and its parameters are equal to those of the *asynchronousQuery* existing method.

2) queryResultsListenerObject

Due to operation of asynchronous queries explained above, to run the new asynchronous query it is necessary also to add a method named *queryResultsListenerObject*, which is executed on the client and which corresponds with *queryResultsListener* existing method. But it gets a list of LOM objects instead of a string as result of the query. Therefore, *queryResults* parameter of *queryResultsListenerObject* method of table I is a list of LOM objects, unlike parameter with the same name of method *queryResultsListener*, which is a String.

IV. ADDITION DOWNLOADRESOURCE METHOD

Actual SQI interface has a set of methods related with connection with a repository or distributed search system, to later search for information on it. However, a last step is needed: once a learning object that matches with parameters

inserted in the search is located, it would be logical to proceed to its download.

The only possibility that exists with actual SQI specification to download a learning object is to create a hyperlink over specified URL, which can exist on the meta-information of learning object. The problem comes when this data is not filled: the learning object could not be downloaded.

Therefore, the second improvement proposed consists in adding new methods that provide the ability to download a resource with its meta-information, whether the metadata that falls within the competence is or not filled. Download will be done synchronously or asynchronously, so a synchronous method (*downloadResource*) and two asynchronous methods (*asynchronousDownloadResource* and *downloadResourceResultsListener*) must be added, for the reasons explained in the previous section; as an asynchronous operation is always the same (first of all, method *setSourceLocation* is invoked, later asynchronous method is invoked and finally, the client invokes *ResultsListener* corresponding method).

A. Addition synchronous downloadResource method

The download using the synchronous method, named *downloadResource*, can be done with an array of bytes or in empty content, in case of using advanced techniques in sending data through SOAP messages. The download in an array of bytes is the most simple and it does not suppose problems of interoperability between systems (this type of data is the most used standard in the representation of physical files and it is present in all programming platforms), but it is less efficient (it blocks the communication flow between client and server until the object has been completely transmitted). However, using advanced techniques of transmission of data through SOAP messages allows transmitting data gradually, without blocking the process in the destination. An existing specification, namely MTOM (SOAP Message Transmission Optimization Mechanism), promotes the use of this type of techniques. Thanks to this specification, the content of a resource could be transmitted out of SOAP message instead of within it, as would be if you used arrays of bytes. Obviously, an agreement in this sense is necessary in order to both parts use the same technique for sending and receiving of data.

The parameters of *downloadResource* method are the following:

- *sessionID*: identifier of session that previously was established with the repository.
- *resourceID*: identifier of the resource you want to download. It permits to identify each resource uniquely within the set of resources that each repository have. This idea is compatible with SPI specification, as identifiers of resources are used in it when publishing.
- *downloadType*: type of download you want to use. Value 0 is used to download it using an array of bytes, value 1 is used to send the file out of SOAP message.

Faults that can be produced by this method are:

- *No such session* (SQI-00013): *sessionID* parameter is not valid.
- *Invalid resource identifier* (SQI-00017): specified identifier does not match any of those that repository manage.
- *Download IO error* (SQI-00018): the download of resource has caused an I/O error.
- *Download Method not supported* (SQI-00019): the download mode is not supported or *downloadType* parameter has an invalid value.
- *Method failure* (SQI-00001): the operation failed by another reason.

B. Addition asynchronous download methods

1) asynchronousDownloadResource method

The asynchronous method, named *asynchronousDownloadResource*, has the same parameters as *asynchronousQuery* and *asynchronousQueryObject* methods.

2) downloadResourceResultsListener method

Finally, with respect to *downloadResourceResultsListener* method, the only thing to note is that *queryResults* parameter is a list of objects of a chosen standard (in this case, LOM, as mentioned above).

V. CONCLUSIONS

The SQI specification has largely facilitated the interoperability between repositories, as well as between them and search engines.

However, there are some areas in which the current specification can be improved. This paper has analyzed some of the failures of it and proposed improvements that could be done to solve them. These proposals were implemented in the LORA-SQI system [6].

SQI query interface has all the features necessary to ensure interoperability between systems, but now, with proposed changes, it will also carry out its task in a structured and efficient manner.

REFERENCES

[1] *SQI: Simple Query Interface*. European Committee for Standarization, 2005. ftp: //ftp.cenorm.be/PUBLIC/CWAs/e-Europe/WS-LT/CWA15454-00-2005-Nov.pdf.

[2] *SPI: Simple Publishing Interface*. European Committee for Standarization, 2008. http: //ariadne.cs.kuleuven.be/lomi/images/b/ba/CEN_SPI_interim_report.pdf.

[3] Hilera, J.R., Otón, S., Ortiz, A., De Marcos, L., Martínez, J.J., Gutiérrez, J.A., Gutiérrez, J.M., Barchino, R. "Evaluating simple query interface compliance in public repositories". *Proceedings of ICALT2009*, IEEE Press, Latvia, 2009, pp. 311-315.

[4] *IEEE 1484.12.1 Learning Object Metadata (LOM)*, Institute of Electrical and Electronics Engineers, 2002.

[5] *Web Services Description Language (WSDL)*, World Wide Web Consortium, 2001. http: //ww w.w3.org/TR/wsdl.

[6] Otón, S., Ortiz, A., Hilera, J.R., Barchino, R., Gutiérrez, J.M., Martínez, J.J., Gutiérrez, J.A., de Marcos, L., Jiménez, L. "Service Oriented Architecture for the Implementation of Distributed Repositories of Learning Objects". *International Jounal of Innovative Computing, Information and Control (IJICIC),* 2010.