World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:13, No:7, 2019

Implementation of a Serializer to Represent PHP Objects in the Extensible Markup Language

Lidia N. Hernández-Piña, Carlos R. Jaimez-González

***Abstract*—**Interoperability in distributed systems is an important feature that refers to the communication of two applications written in different programming languages. This paper presents a serializer and a de-serializer of PHP objects to and from XML, which is an independent library written in the PHP programming language. The XML generated by this serializer is independent of the programming language, and can be used by other existing Web Objects in XML (WOX) serializers and de-serializers, which allow interoperability with other object-oriented programming languages.

***Keywords*—**Interoperability, PHP object serialization, PHP to XML, web objects in XML, WOX.

I. Introduction

THE growth in the use of information and communication technologies has originated the development of applications written in different programming languages, which at the same time has generated interoperability problems when interchanging information among them. One way to solve interoperability among object-oriented programming languages is through serialization of objects, where the issues are related to data type mapping, representation of objects, messages, and serialization and de-serialization processes, as it is stated in [1].

Concerning data type mapping, "data types are one of the main issues when it comes to interoperability between different programming languages. There must be an agreed mapping between the data types of the programming languages involved. One way to solve this problem is a mapping table with the different data types supported by the different programming languages" [1].

In order to tackle the problem of object representation, "there must be a standard way of representing objects, either the object is written in Java, C#, PHP, or other object-oriented programming language. A standard format must be established to represent the supported structures in the different programming languages: classes, primitive data types, arrays, and user-defined classes" [1].

The issue about messages is that "they represent the way clients and servers communicate. Messages are used to make requests or receive responses, and they must also be written in a standard way to be understood by clients and servers" [1].

Finally, regarding serialization and de-serialization, "in the context of data storage and transmission, serialization is the process of rendering an object into a state that can be saved persistently into a storage medium, such as a file, database, or a stream to be transmitted through the network. De-serialization is the opposite process, which puts the serialized version of the object into a live object in memory" [1].

This paper presents a serializer and a de-serializer of PHP objects to XML, called PHP Web Objects in XML (PHPWOX) [2]. The XML generated by PHPWOX is independent of the programming language, and can be used by other existing serializers and de-serializers WOX [3], which allow interoperability between applications written in PHP and applications written in the programming languages supported by WOX: Java, C# [1] and Python [4].

The rest of the paper is organized as follows. Section II presents concepts and some existing PHP parsers and serializers. Section III provides an explanation of the classes that are part of PHPWOX; it also presents its architecture and a description of the functionality of the modules that compose it. Finally, conclusions and future work are provided in Section IV.

II. Concepts and Existing PHP Serializers

This section provides some concepts used for understanding the functionality of the object serializers to XML, such as serialization and de-serialization. It also describes some existing PHP parsers and serializers, and a comparison among them is provided.

A. Concepts

Serialization is the process of rendering an object to a state that can be stored permanently to a medium, such as a file, with the aim of transmitting it through the network as a series of bytes, or in other format, such as XML or JSON. The serialization of objects to XML provides a representation that is understandable by a human and by a computer; this format also promotes interoperability among different programming languages. The series of bytes or the format chosen to serialize can be used to reconstruct the object, which will be identical to the original.

The program able to serialize an object directly to a file in an automatic way is called a serializer; while a parser is just a program that reads and writes XML to a file, in which the serialization is done manually by the user.

Fig. 1 shows the process to serialize an object to XML, which starts by obtaining the name, type and value of each attribute of the object, through the use of reflection. Reflection is the ability of a program of observing and optionally

Lidia N. Hernández-Piña is with the Department of Information Technology at the Metropolitan Autonomous University, Mexico City, Mexico (e-mail: 210368177@alumnos.cua.uam.mx)..

Carlos R. Jaimez-González is with the Department of Information Technology at the Metropolitan Autonomous University, Mexico City, Mexico (corresponding author, e-mail: cjaimez@correo.cua.uam.mx).

modifying the high level structure of an object; this ability allows accessing the information of objects, knowing their attributes and public methods in execution time. Introspection is also used in order to obtain the data type of a specific attribute. The second step in the process is to write an XML file with a tree structure that represents the object in XML, taking into consideration for each object all its attributes; and in case of having nested objects, doing it recursively.

The process of de-serializing an object from XML is illustrated in Fig. 2. It first extracts from the XML file the object information; then an object is created with that information, which is obtained from the XML file. In case the object does not exist, it is created first with all the attributes needed.

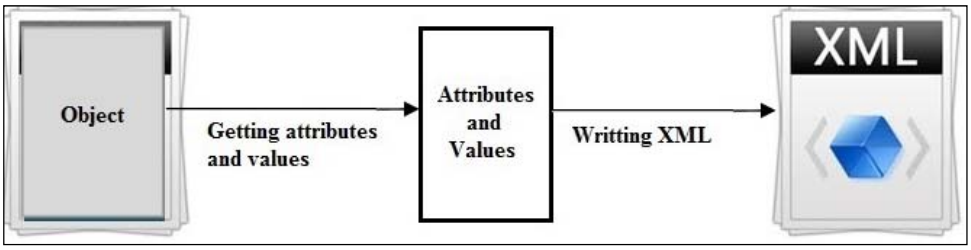

Fig. 1 Process of serializing an object to XML

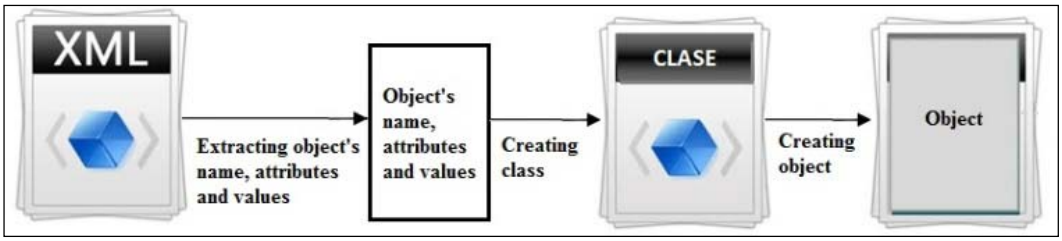

Fig. 2 Process of de-serializing an object from XML

B. Existing Parsers and Serializers

Some of the existing PHP parsers and serializers were analyzed and compared. The parsers taken into consideration were *DOM* and *SimpleXML*; the non-XML serializers analyzed were the *serialize()* function of PHP, the *json_encode* function of PHP, and *Igbinary*; and the only XML serializer analyzed was *Pear*. Although there are several existing parsers and serializers of PHP objects, incorporated as PHP functions or as external complements, none of them allow interoperability with other programming languages. In the following paragraphs, a brief description of each parser or serializer is provided.

Document Object Model (DOM) [5]. DOM is an application programming interface (API) that provides a standard set of objects to represent HTML and XML documents; it is a model to describe how to combine such objects; and it is a standard interface to access and manipulate them. Programs can access and modify content, structure, and style of HTML and XML documents through DOM [1]. PHP has a DOM extension, included when downloading and installing PHP with XAMPP [6], which allows creating and manipulating XML documents. It should be noted that with DOM, the programmer must specify how every object should be serialized, which means that there is no way to automatically serialize any objects of any class; this process is carried out manually by the programmer.

SimpleXML [7]. It is a PHP extension included when downloading and installing PHP with XAMPP, which allows manipulating XML files. SimpleXML allows to process XML files with property selectors and matrix iterators. Compared to DOM, SimpleXML takes less code to be written in order to read an element. The programmer must specify how every object should be serialized with SimpleXML, which means that there is no way to automatically serialize any objects of any class; this process is also carried out manually by the programmer.

Serialize() function [8]. This function of PHP returns a string that contains a stream of bytes that represent any value that can be stored in PHP; this means that the serialization is not an XML representation. The *unserialize()* function can restore the original values from the string mentioned. Using the *serialize()* function to store an object, it will store all variables of the object, which represents its state. For this type of serialization, it is needed the class from which the object was created.

Json_encode() function [9]. This function returns a value serialized in a string. PHP implements a superset of JSON, and also codes and decodes scalar values and NULL values. JSON standard only accepts these values when they are nested in an array or in an object. The *json_encode()* function can serialize objects directly only if its attributes are public; if they are private, there is no standard way to serialize it.

IgBinary [10]. This serializer replaces the standard PHP serializer. It does not use a text representation, but a binary representation instead. A clear advantage of this serializer is its little use of memory, but on the other hand, the object representation is not visible to the user.

Pear XML_Serialize [11]. Pear is a framework for reusable

PHP components. *XML_Serialize* is part of this framework and allows to serialize complex data to XML, such as arrays and objects. Pear allows a direct (automatic) serialization, which means that the user does not need to specify how the serialization takes place.

C. Comparison of Existing Parsers and Serializers

Table I shows a comparison of features among the parsers and serializers mentioned in the previous sections: P1) DOM, P2) SimpleXML, P3) Serialize(), P4) Json_encode(), P5) IgBinary, P6) XML_Serializer; the tick indicates that the parser or serializer has the feature, and the cross indicates that the parser or serializer does not have it. A brief description of the features considered, which were used to evaluate the existing parsers and serializers, is presented in the following paragraphs.

Interoperability. It is the possibility of serializing an object in a programming language, and de-serializing it in a different programming language, and vice versa. For example, serializing an object in the C# programming language, and de-serializing it in the Python programming language.

Convert objects-XML. This feature refers to the possibility of serializing an object automatically to an XML file, through a method of a class.

Convert XML-objects. This feature refers to the possibility of obtaining one or more objects automatically from an XML file, through a method of a class.

Free license. The parser or serializer is distributed freely, it is available for use and for unlimited time.

Well formed XML. This feature means to have a document in which its tags are correctly nested, and also the corresponding initial and end tags.

Documentation. This feature means that the parser or serializer has documentation and a user guide for its use.

Examples of use. This feature means that the parser or serializer has examples of use in the documentation or in any other source of information.

TABLE I
FEATURES TO EVALUATE THE PARSERS AND SERIALIZERS ANALYZED

Feature	*P1*	*P2*	*P3*	*P4*	*P5*	*P6*
Interoperability	✗	✗	✗	✓	✗	✗
Convert objects-XML	✗	✗	✗	✗	✗	✗
Convert XML-objects	✗	✗	✗	✗	✗	✗
Free license	✓	✓	✓	✓	✓	✓
Well formed XML	✓	✓	✗	✗	✗	✓
Documentation	✓	✓	✓	✓	✓	✓
Examples of use	✓	✓	✓	✓	✓	✓

III. IMPLEMENTATION OF THE XML SERIALIZER

This section provides an explanation of the classes that are part of the serializer and de-serializer; it also presents its architecture and a description of the functionality of the modules that compose it.

A. Classes Implemented

Fig. 3 shows a diagram with the classes that are part of the XML serializer and de-serializer, which are the following: *Easy* class, *SimpleWriter* class, *SimpleReader* class, *Encode* class, *Serial* interface, *CreateClass* class, *TypeMapping* class.

Easy. This is an external class, used by final users in order to serialize and de-serialize objects to and from XML. It has two methods: *save(ob:object, filename:file)*, which serializes an object to XML and saves it to an XML file, the first parameter is the object to be serialized, and the second parameter is the file to store the XML generated; and the *load(filename:file):Object*, which de-serializes an object from the XML file that is specified in the input parameter.

SimpleWriter. This class takes a PHP object and represents it as XML by using the XML DOM parser. It uses a *write()* method that receives a PHP object, analyzes it to determine its data type, extracts its fields and values, and serializes it.

SimpleReader. This class represents the de-serializer; it reads an object from an XML file to represent it as a PHP object. It uses the *read()* method to take as starting point a DOM element from the XML file, determines its data type, and extracts its information in order to create the specific object required. It should be noted that if the class of the object does not exist, it uses the *createClass* class to create it.

Encode. This class allows to encode and decode the base64 matrixes of bytes; it has several methods to take care of the encode and decode operations, two of them are: *encode(byte[] source)*, and *decode(byte[] source).*

TypeMapping. This class provides the data type mappings from the PHP programming language to WOX, and vice versa.

Serial. This is an interface that defines the constants used in the XML representation of PHP objects.

CreateClass. This is a class that generates classes; it allows to create a PHP document with a class, taking as input its name and attributes. The generated class will have a constructor, setters and getters.

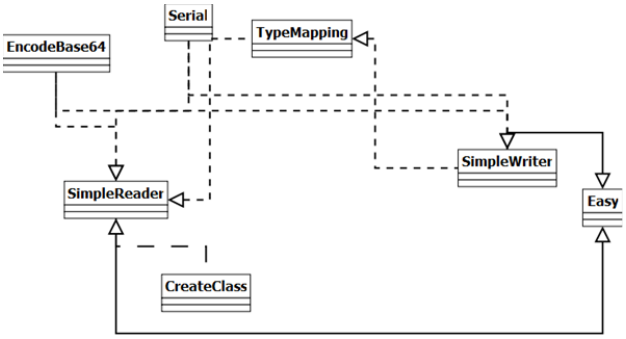

Fig. 3 Classes implemented for the serializer and de-serializer

B. Architecture and Modules

The main modules of PHPWOX are the following: 1) serializer module, which serializes PHP objects to XML (*SimpleWriter* class); 2) de-serializer module, which de-serializes PHP objects from XML (*SimpleReader* class); and 3) generator classes module, which creates PHP classes from XML (*CreateClass* class). Fig. 4 shows a diagram with the functionality of the modules. There are some notes indicated as A1, A2, A3, A4 and A5, which are explained in the following paragraphs.

A1. In this section it is observed the *ProductJava* object of the *Product* class, which is written in the Java programming language, with its attributes and values. This object is serialized through the Java WOX serializer [1], which gives the *ProductJava.xml* file shown to the right.

A2. The *ProductJava.xml* file is de-serialized with the PHP WOX de-serializer [2], which does not have the *Product* class, so it creates the *Product.php* class, and creates an instance of this class with the values indicated in the *JavaProduct.xml* file, giving as result the *ProductPhp* object in the PHP programming language, which is identical to the *ProductJava* object, but in PHP.

A3. The *ProductPhp* object gets new values in its attributes.

A4. The *ProductPhp* object is serialized by PHP WOX Serializer [2], obtaining the *ProductPhp.xml* file.

A5. The *ProductPhp.xml* file is de-serialized by the Java WOX de-serializer [1], obtaining as result the object modified in PHP. In Java, this object is called *ProductJavaMod*, which is shown in Fig.4.

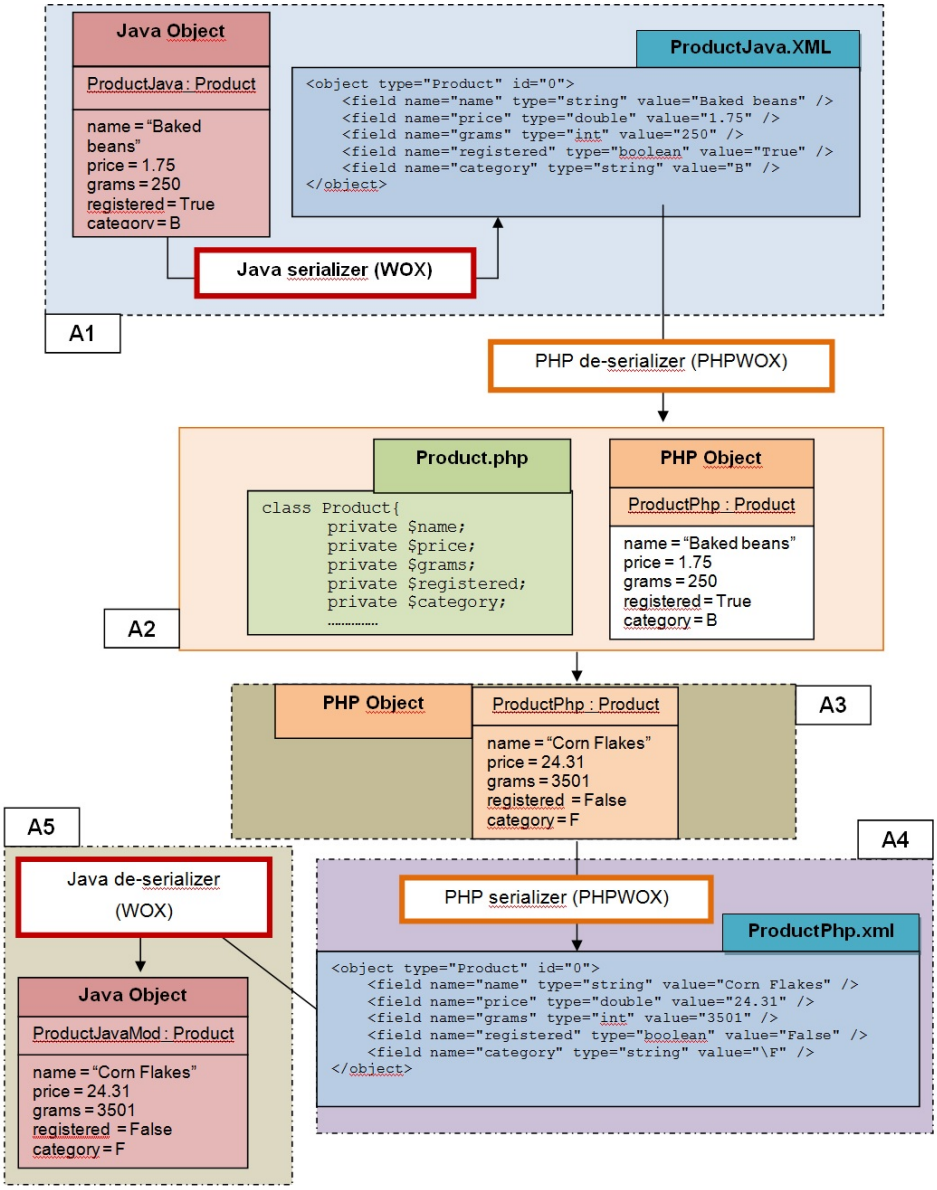

Fig. 4 Diagram with the modules for the serializer and de-serializer

IV. Conclusions and Future Work

This paper presented a serializer and de-serializer of PHP objects to and from XML, called PHPWOX. The XML generated by PHPWOX is independent of the programming language, and can be used by other existing WOX serializers and de-serializers, which allow interoperability with the object-oriented programming languages Java, C#, and Python. The development of PHPWOX was described, the set of classes implemented, and the architecture and modules were also presented, with a clear example of the functionality of the modules, and showing what PHPWOX is capable of doing

regarding interoperability.

Further work is needed to implement a framework on top of PHPWOX, to handle distributed objects, in order to communicate with other WOX frameworks already implemented.

REFERENCES

[1] C. Jaimez-González, S. M. Lucas, "Easy Serialization of C# and Java Objects", Proceedings of the Balisage: The Markup Conference 2011, ISSN: 1947-2609, Montreál, Canada, 2-5 August 2011, http://www.balisage.net/Proceedings/vol7/html/Jaimez01/BalisageVol7-Jaimez01.html, last access in August 2017.

[2] PHPWOX web site, http://phpwoxserializer.sourceforge.net/, last access in August 2017.

[3] WOX web site, http://woxserializer.sourceforge.net/, last access in August 2017.

[4] A. Rodríguez-Martínez, C. Jaimez-González, "Serializador de Objetos a XML en el Lenguaje de Programación Python", Avances de Ingeniería Electrónica 2013, Universidad Autónoma Metropolitana, Universidad Autónoma de Nayarit, México, 2013.

[5] DOM reference web site, https://www.w3.org/DOM/, last access in August 2017.

[6] XAMPP web site, http://www.apachefriends.org/es/download.html, last access in August 2017.

[7] SimpleXML documentation web site, http://www.php.net/manual/en/book.simplexml.php, last access in August 2017.

[8] Serialize function documentation web site, http://php.net/manual/es/language.oop5.serialization.php, last access in August 2017.

[9] json_encode function documentation web site, http://us2.php.net/manual/es/function.json-encode.php, last access in August 2017.

[10] Igbinary serializer web site, http://pecl.php.net/package/igbinary, last access in August 2017.

[11] Pear XML Serializer web site, http://pear.php.net/, last access in August 2017.