ISSN 1870-4069

Serialización de objetos PHP a XML

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

Universidad Autónoma Metropolitana, Unidad Cuajimalpa,
Departamento de Tecnologías de la Información, México

210368177@alumnos.cua.uam.mx, cjaimez@correo.cua.uam.mx

Resumen. La serialización es el proceso de escribir un objeto en un medio de almacenamiento como un archivo o un buffer de memoria, con el objetivo de transmitirlo a través de una red; la deserialización es el proceso inverso. En este artículo se presenta un serializador y deserializador de objetos PHP a XML, el cual es una biblioteca independiente escrita en el lenguaje de programación PHP. Una de sus principales características es la generación de representaciones XML estándar, las cuales son independientes del lenguaje de programación e interoperables con los serializadores *Web Objects in XML* (WOX) existentes.

Palabras clave: Serialización de objetos PHP, web objects in XML, PHP a XML, interoperabilidad, WOX.

Serialization of PHP Objects to XML

Abstract. Serialization is the process of writing an object to a storage medium as a file or as a buffer in memory, with the aim of transmitting it through a network; de-serialization is the inverse process. This paper presents a serializer and de-serializer of PHP objects to XML, which is an independent library written in the PHP programming language. One of its main features is the generation of standard XML representations, which are independent of the programming language and interoperable with the existing *Web Objects in XML* (WOX) serializers.

Keywords. Serialization of PHP objects, web objects in XML, PHP to XML, interoperability, WOX.

1. Introducción

La interoperabilidad es una característica importante en los sistemas distribuidos basados en objetos, ya que permite la comunicación de programas (clientes y servidores), escritos en diferentes lenguajes de programación orientados a objetos. Existen algunos problemas fundamentales que tienen que ser resueltos por los diferentes len-

guajes para alcanzar la interoperabilidad. Algunos de estos problemas se exponen a continuación.

Mapeo de tipos de datos. Los tipos de datos son uno de los principales problemas para alcanzar la interoperabilidad entre diferentes lenguajes de programación. Debe de existir un acuerdo de mapeo entre los tipos de datos de los lenguajes de programación que se desean comunicar. Una forma de resolver este problema es mediante tablas de mapeo con los diferentes tipos de datos soportados por los lenguajes de programación que desean comunicarse.

Representación de objetos. Se debe establecer una forma estándar de representar objetos, ya sea que estén escritos en Java, C#, C++, o algún otro lenguaje de programación orientado a objetos. El formato estándar debe considerar la representación de las estructuras y tipos de los diferentes lenguajes de programación, tales como clases, tipos de datos primitivos, arreglos, y clases definidas por el usuario.

Mensajes. Los mensajes que se intercambian también deben de estar escritos en una forma estándar para que puedan ser entendidos por ambas partes.

Serialización y deserialización. En el contexto de almacenamiento y transmisión de datos, la serialización es el proceso de transformar un objeto a un estado en el que pueda ser almacenado permanentemente a un medio tal como un archivo, una base de datos, o un flujo para ser transmitido a través de la red. Deserialización es el proceso inverso, en el cual se transforma la versión serializada en XML del objeto en un objeto en memoria.

Este artículo presenta un serializador y deserializador de objetos PHP a XML, llamado *PHP Web Objects in XML* (PHPWOX) [1]. El XML generado por PHPWOX es independiente del lenguaje de programación, y puede ser utilizado por otros serializadores y deserializadores *Web Objects in XML* (WOX) [2], los cuales permiten interoperabilidad entre aplicaciones escritas en PHP y aplicaciones escritas en los lenguajes de programación soportados por WOX: Java, C# [3] y Python [4].

El resto del artículo está organizado de la siguiente manera. La sección 2 proporciona algunos conceptos importantes para entender los procesos de serialización y deserialización. En la sección 3 se presentan algunos parsers y serializadores PHP que fueron analizados. Una tabla comparativa de parsers y serializadores se presenta en la sección 4. La sección 5 describe las clases que son parte de PHPWOX, se presenta su arquitectura y una descripción de la funcionalidad de los módulos que lo componen. Finalmente, en la sección 6 se presentan las conclusiones y el trabajo a futuro.

2. Conceptos

La serialización es un proceso que consiste en la codificación de un objeto en un medio de almacenamiento como puede ser un archivo o un buffer de memoria, con el objetivo de transmitirlo a través de una conexión en red como una serie de bytes o en otro formato como XML. La serie de bytes o el formato pueden ser usados para crear un nuevo objeto que es idéntico en todo al original, incluido su estado interno.

Al programa capaz de serializar un objeto directamente en un archivo de forma automática se le denomina *serializador*; mientras que un *parser* es un programa que

puede leer y escribir XML en un archivo, en éste la serialización es de forma personalizada, es decir, la realiza el usuario del *parser*.

Para serializar objetos en XML básicamente se sigue el proceso que se observa gráficamente en la Figura 1, el cual se describe a continuación.

1. Se obtiene el nombre, tipo y valor de cada uno de los atributos del objeto. Esto se realiza mediante el proceso de reflexión, el cual es la capacidad que tiene un programa para observar y opcionalmente modificar su estructura de alto nivel. Por medio de esta capacidad es posible acceder a la información de los objetos, conociendo y/o ejecutando sus atributos y métodos públicos, todo ello en tiempo de ejecución; también se utiliza la introspección, la cual es la obtención del tipo de dato en particular.
2. Una vez que se obtiene el nombre y valor de cada atributo del objeto a serializar, se escriben en un archivo XML. En caso de que el valor no sea tipo primitivo sino un objeto, habrá que representar también en el documento XML todos los atributos de dicho objeto.

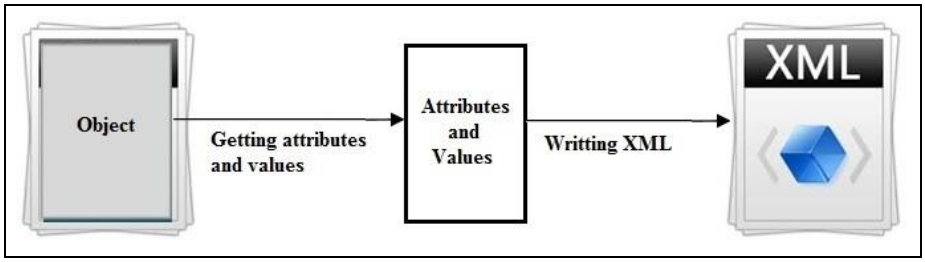

Fig. 1. Serialización de un objeto a XML.

El proceso inverso es la deserialización, en la cual se siguen los siguientes pasos y se puede observar en la Figura 2.

1. Se extrae la información acerca de un objeto del archivo XML.
2. Un objeto es una instancia de una clase, por lo cual para crear el objeto es necesario primero crear una clase basándose en la información obtenida del archivo XML, por lo que se crea la clase.
3. Se instancia el objeto con la información obtenida del documento XML.

3. Parsers y serializadores existentes

En esta sección se presenta el análisis y comparación de algunos *parsers* y serializadores PHP existentes. Los *parsers* que se analizaron fueron DOM y *SimpleXML*; los serializadores que no utilizan XML que se revisaron fueron la función *serialize()* de PHP, la función *json_encode()* de PHP y la biblioteca *Igbinary*; y el único serializador XML que se analizó fue *Pear*. Aunque existen varios *parsers* y serializadores de objetos PHP incorporados como funciones PHP o como complementos externos, ninguno de ellos permite interoperar con otros lenguajes de programación. En los

siguientes párrafos se proporciona una breve descripción de cada *parser* o serializador.

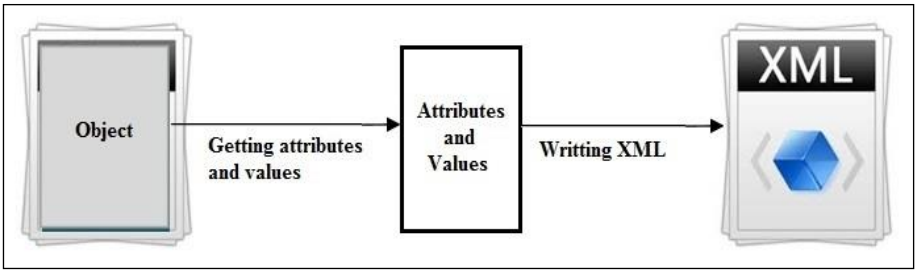

Fig. 2. Deserialización de un objeto a partir de un documento XML.

Document Object Model (DOM) [5]. El modelo de objetos de documento es una interfaz de programación de aplicaciones (*Application Programming Interface*, API por sus siglas en inglés) que proporciona un conjunto estándar de objetos para representar documentos HTML y XML, es un modelo para describir cómo combinar tales objetos, y es una interfaz estándar para accederlos y manipularlos. A través de DOM es posible acceder y modificar el contenido, estructura y estilo de documentos HTML y XML. PHP tiene una extensión de DOM incluida en la distribución de XAMPP [6], la cual permite crear y manipular documentos XML. Cabe señalar que con DOM el programador debe especificar cómo debe serializarse cada objeto, lo cual significa que no hay forma para serializar automáticamente los objetos de cualquier clase; este proceso es llevado a cabo manualmente por el programador.

SimpleXML [7]. Es una biblioteca de PHP incluida en la distribución de XAMPP, la cual permite manipular archivos XML. Comparado con *DOM*, *SimpleXML* requiere la escritura de menos código para leer elementos de un documento XML. Al igual que con *DOM*, una desventaja de *SimpleXML* es que el programador debe especificar cómo debe serializarse cada objeto, este proceso es llevado a cabo manualmente por el programador.

Función serialize() [8]. Es una función de PHP que regresa una cadena que contiene un flujo de *bytes* que representan cualquier valor que puede ser almacenado en PHP, lo cual significa que la serialización no es una representación en XML. La función *unserialize()* puede restaurar los valores originales a partir de la cadena mencionada. Utilizando la función *serialize()* para almacenar un objeto se guardarán todos los valores de las variables del objeto. Para este tipo de serialización es necesario tener la clase a partir de la cual el objeto fue creado.

Función json_encode() [9]. Esta función regresa un valor serializado en una cadena. PHP implementa un conjunto más amplio de JSON y también codifica y decodifica valores escalares y valores nulos. El estándar de JSON solamente acepta estos valores cuando están anidados en un arreglo o en un objeto. La función *json_encode()* puede serializar objetos directamente solo si sus atributos son públicos.

IgBinary [10]. Es un serializador que sustituye al serializador predeterminado de PHP. Este serializador no utiliza una representación de texto, sino una representación

binaria. Una ventaja de este serializador es que utiliza muy poca memoria para la representación de un objeto; sin embargo, una desventaja es que la representación del objeto no es visible al usuario.

Pear XML_Serialize [11]. Es un *framework* de componentes PHP reutilizables. *XML Serialize* es parte de este *framework* y permite serializar datos complejos a XML, tales como arreglos y objetos. *Pear* permite serialización directa de un objeto a XML, lo cual significa que el usuario no necesita especificar cómo se debe realizar la serialización del objeto.

4. Comparación

La Tabla 1 muestra una comparación de las características que se tomaron en cuenta en los *parsers* y serializadores analizados: H1) *DOM*, H2) *SimpleXML*, H3) Función *serialize()*, H4) Función *json_encode()*, H5) *IgBinary*, H6) *Pear XML_Serializer*. El símbolo de verificación indica que el *parser* o serializador tiene la característica que se indica, mientras que la *x* indica que no la tiene. Una breve descripción de las características consideradas para evaluar los *parsers* y serializadores se presenta en los siguientes párrafos.

Tabla 1. Comparación de características de los parsers y serializadores analizados.

Características	*H1*	*H2*	*H3*	*H4*	*H5*	*H6*
Interoperabilidad	×	×	×	✓	×	×
Convertir objeto a XML	×	×	×	×	×	×
Convertir XML a objeto	×	×	×	×	×	×
Licencia gratuita	✓	✓	✓	✓	✓	✓
XML bien formado	✓	✓	×	×	×	✓
Manual usuario/documentación	✓	✓	✓	✓	✓	✓
Ejemplos de uso	✓	✓	✓	✓	✓	✓

Interoperabilidad. Es la posibilidad de serializar un objeto en un lenguaje de programación y deserializarlo en un lenguaje distinto, y viceversa. Por ejemplo, serializar un objeto en el lenguaje PHP y deserializarlo en el lenguaje Java.

Convertir directamente objeto a XML. Es la posibilidad de serializar un objeto a XML automáticamente por medio de un método o procedimiento existente, sin necesidad de que el programador lo tenga que hacer manualmente.

Convertir directamente XML a objeto. Es la posibilidad de obtener uno o más objetos automáticamente a partir de un método o procedimiento existente y un archivo XML, sin necesidad de que el programador lo tenga que hacer manualmente.

Licencia gratuita. Se refiere a que el serializador se distribuye sin costo, disponible para su uso y por tiempo ilimitado.

XML bien formado. Se refiere a un documento XML en el cual las etiquetas están correctamente anidadas, además de tener sus correspondientes etiquetas iniciales y finales.

Manual de usuario y documentación. Se refiere a la existencia de un manual o guía de uso del *parser* o serializador.

Ejemplos de uso. Es la presencia de ejemplos de uso dentro de la documentación y/o manual del serializador o *parser* a analizar.

5. Implementación del serializador

Esta sección proporciona una explicación de las clases que son parte del serializador y del deserializador que se desarrolló, también se presenta su arquitectura y una descripción de la funcionalidad de los módulos principales que la componen.

5.1. Clases

La Figura 3 muestra un diagrama con las clases que fueron desarrolladas como parte del serializador y del deserializador, las cuales son las siguientes: *Easy*, *SimpleWriter*, *SimpleReader*, *Encode*, *Serial*, *CreateClass* y *TypeMapping*.

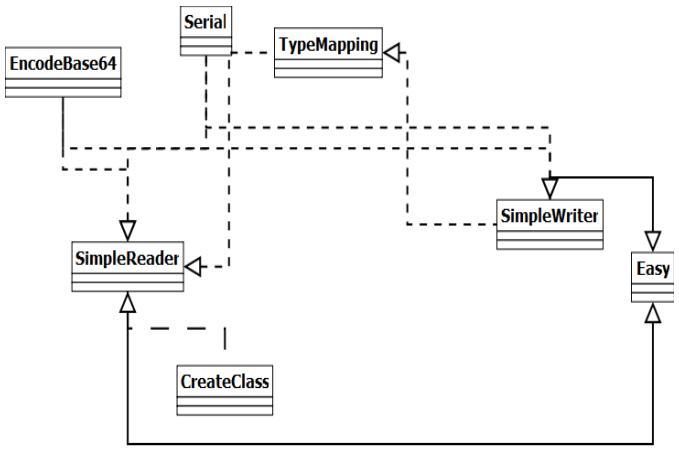

Fig. 3. Clases implementadas para el serializador y para el deserializador.

Easy. Esta clase es utilizada por los usuarios finales para serializar y deserializar objetos a y desde XML. Tiene dos métodos: *save (ob:object, filename:file)*, el cual serializa un objeto a XML y lo almacena a un archivo XML, el primer parámetro es el objeto a ser serializado, y el segundo parámetro es el archivo para almacenar el XML generado; y el método *load (filename:file):Object*, el cual deserializa un objeto desde el archivo XML que es especificado en el parámetro de entrada.

SimpleWriter. Esta clase toma un objeto PHP y lo representa como XML, utilizando el *parser* XML DOM. Utiliza un método *write()* que recibe un objeto PHP, lo analiza para determinar su tipo de dato, extrae sus campos y valores, y lo serializa.

SimpleReader. Esta clase representa el deserializador, lee un objeto de un archivo XML para representarlo como un objeto PHP. Esta clase utiliza el método *read()* para

tomar como punto de partida un elemento DOM del archivo XML, determina su tipo de dato, y extrae su información para crear el objeto específico requerido. Cabe señalar que si la clase del objeto no existe, utiliza la clase *createClass* para crearla.

Encode. Esta clase permite codificar y decodificar matrices de *bytes* base64; tiene varios métodos para realizar las operaciones *encode()* y *decode()*, tales como los siguientes: *encode (byte[] source)*, y *decode (byte[] source)*.

TypeMapping. Esta clase proporciona el mapeo de tipos de datos del lenguaje de programación PHP a WOX y viceversa.

Serial. Interfaz que define constantes para la representación XML de objetos PHP.

CreateClass. Esta clase genera clases; permite crear un documento PHP con una clase, tomando como entrada su nombre y sus atributos. La clase generada tendrá un constructor, métodos *set* y *get* para cada uno de sus atributos.

6. Arquitectura y módulos

Los módulos principales de PHPWOX son los siguientes: 1) módulo serializador, el cual serializa objetos PHP a XML a través de la clase *SimpleWriter*; 2) módulo deserializador, el cual deserializa objetos PHP desde un documento XML a través de la clase *SimpleReader*; y 3) módulo generador de clases, el cual crea clases PHP a partir del XML mediante la clase *CreateClass*. La Figura 4 muestra un diagrama con la funcionalidad de los módulos del serializador y deserializador desarrollados, en donde se muestran los objetos y sus representaciones XML después de haber sido serializados y deserializados. Hay algunas notas indicadas como A1, A2, A3, A4 y A5, las cuales son explicadas en los siguientes párrafos.

A1. En esta sección se observa el objeto *ProductJava* de la clase *Product*, el cual está escrito en el lenguaje de programación Java, con sus atributos y valores. Este objeto es serializado a través del serializador WOX [2, 3], el cual genera el archivo *ProductJava.xml* mostrado a la derecha.

A2. El archivo *ProductJava.xml* es deserializado con el deserializador PHPWOX [1], el cual no tiene la clase *Product*, de tal forma que crea la clase *Product.php*, y crea una instancia de esa clase con los valores indicados en el archivo *JavaProduct.xml*, dando como resultado el objeto *ProductPhp* en el lenguaje de programación PHP, el cual es idéntico al objeto *ProductJava*, pero en PHP.

A3. El objeto *ProductPhp* obtiene nuevos valores en sus atributos.

A4. El objeto *ProductPhp* es serializado por el serializador PHPWOX, obteniendo el archivo *ProductPhp.xml*.

A5. El archivo *ProductPhp.xml* es deserializado por el deserializador WOX en Java [2], obteniendo como resultado el objeto modificado en PHP. En Java este objeto es llamado *ProductJavaMod*, el cual es mostrado en la Figura 4.

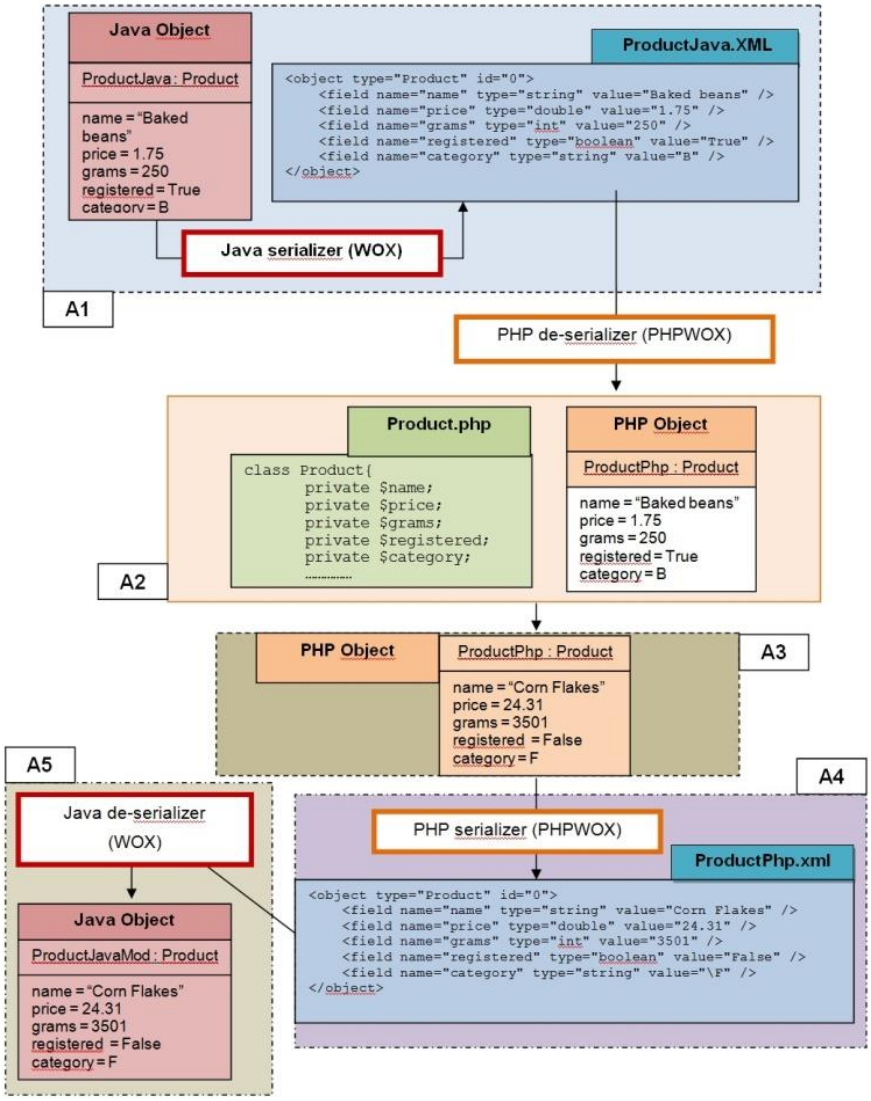

Fig. 4. Diagrama con los módulos para el serializador y deserializador.

7. Conclusiones y trabajo futuro

Este artículo presentó un serializador y deserializador de objetos PHP a XML, llamado PHPWOX. Se puede concluir que el XML generado por PHPWOX es independiente del lenguaje de programación, y puede ser utilizado por otros serializadores y deserializadores WOX existentes, lo cual permite interoperabilidad con los lenguajes de programación Java, C# y Python.

El desarrollo de PHPWOX fue descrito, el conjunto de clases implementadas, la arquitectura y los módulos, con un claro ejemplo de la funcionalidad de los módulos, y mostrando las capacidades de PHPWOX. En el ejemplo presentado se evidencia la interoperabilidad lograda entre PHPWOX y los demás serializadores WOX, ya que se muestran los objetos y sus correspondientes representaciones XML, las cuales fueron obtenidas al utilizar el serializador y deserializador PHPWOX.

Como parte del trabajo futuro es necesario implementar un *framework* por encima de PHPWOX, el cual permita manejar objetos distribuidos, para poder comunicarse con otros *frameworks* WOX que ya están implementados.

Referencias

1. PHPWOX. Disponible en: http://phpwoxserializer.sourceforge.net/
2. Serializadores Web Objects in XML (WOX). Disponible en: http://woxserializer.sourceforge.net/
3. Jaimez-González, C., Lucas, S., López-Ornelas, E.: Easy XML Serialization of C# and Java Objects. In: Proceedings of the Balisage. The Markup Conference 2011, Balisage Series on Markup Technologies, 7 (2011)
4. Rodríguez-Martínez, A., Jaimez-González, C. R.: Serializador de Objetos a XML en el Lenguaje de Programación Python. Avances de Ingeniería Electrónica 2013, 444–451 (2013)
5. Document Object Model (DOM) para PHP. Disponible en: http://php.net/manual/es/book.dom.php
6. XAMPP. Disponible en: http://www.apachefriends.org/es/download.html
7. SimpleXML. Disponible en: http://www.php.net/manual/en/book.simplexml.php
8. Serialización de objetos (función serialize). Disponible en: http://php.net/manual/es/language.oop5.serialization.php
9. json_encode. Disponible en: http://us2.php.net/manual/es/function.json-encode.php
10. Igbinary. Disponible en: http://pecl.php.net/package/igbinary
11. Pear XML_Serializer. Disponible en: http://pear.php.net/package/XML_Serializer/redirected