Prácticas

**Estilo:** Normal **Tamaño letra:** Normal **Interlineado:** 1.2

# Desarrollo de Aplicaciones Web

Práctica 12: PHP 6 (DOM)

## 1. Objetivos

- Aprender a manejar el DOM para crear un documento XML.

## 2. Recursos

¿Qué es XML?

- **Extensible Markup Language (XML) 1.0** (1) : especificación oficial de XML según el W3C (*World Wide Web Consortium*).
- **MOOC: Tecnologías de intercambio/integración de datos) 1.0** (2) : curso dedicado al manejo de las principales tecnologias de intercambio e integración de datos que se emplean en la actualidad, como XML, JSON, etc..

¿Qué es el DOM?

- **W3C Documento Object Model (DOM)** (3) : especificación oficial del DOM según el W3C.

¿Cómo puedo manejar el DOM desde PHP?

- **Modelo de Objetos de Documento** (4) : permite manipular el DOM de un documento XML o de una página web.

¿Cómo puedo comprobar que el formato de mi canal web es correcto?

- **W3C Feed Validation Service** (5) : validador gratuito que comprueba si un canal web cumple el formato RSS o Atom.

## 3. ¿Qué tengo que hacer?

En esta práctica tienes que crear una fuente o canal web (*feed*), **con formatos RSS 2.0 y Atom**, donde se muestren los datos de las últimas cinco fotos publicadas. Este canal web recibirá un parámetro que indicará el formato deseado (RSS o Atom). Para crear el canal web se tiene que emplear el DOM de PHP y se tienen que crear y manipular los nodos del documento (no vale crear el documento de salida a partir de una cadena de texto en la que se van concatenando las diferentes partes que componen el documento).

Además, también tienes que añadir la posibilidad de que cada usuario genere una exportación de toda la información que el sistema posee sobre el usuario a un fichero XML. Otra vez, el documento XML se debe crear nodo a nodo mediante el DOM.

El formato del fichero de exportación es libre, su contenido puede ser algo parecido a (es solo una sugerencia):

```
<?xml version="1.0" encoding="UTF-8" ?>
<PI>
<Usuario IdUsuario="1">
      <NomUsuario>Sergio</NomUsuario>
      <Email>sergio.lujan@ua.es</Email>
      <!-- Resto de campos -->

      <Albumes>
            <Album IdAlbum="123">
                  <Titulo>Las vacaciones</Titulo>
                  <Descripcion>Las fotos de las vacaciones en Alicante</Descripcion>
                  <Fotos>
                        <Foto IdFoto="11">
                              <Titulo>Una foto</Titulo>
                              <Descripcion>La descripción de la foto</Descripcion>
                              <!-- Resto de campos -->
                        </Foto>

                        <Foto IdFoto="23">
                              <Titulo>Otra foto</Titulo>
                              <Descripcion>La descripción de la otra foto</Descripcion>
                              <!-- Resto de campos -->
                        </Foto>
                  </Fotos>
                  <Solicitudes>
                        <Solicitud IdSolicitud="2">
                              <Nombre>Sergio Luján Mora</Nombre>
                              <Titulo></Titulo>
                              <!-- Resto de campos -->
                        </Solicitud>
                  </Solicitudes>
            </Album>
      </Albumes>
</Usuario>
</PI>
```

# 4. ¿Cómo lo hago?

## 4.1. Canal RSS

Una fuente o canal web (*feed*) es un mecanismo que permite la redifusión del contenido web. Un canal web permite suministrar información actualizada frecuentemente a los usuarios, llamados suscriptores. Mediante un programa compatible con un canal web, un suscriptor puede recibir una alerta cuando se produzca una actualización en el canal web (y por tanto, en el sitio web de origen de los contenidos).

Existen varios formatos de redifusión de contenido, pero los dos más utilizados son RSS y Atom. Ambos formatos utilizan el lenguaje XML y se suelen identificar en una página web con el icono que aparece en la Figura 1.

**Figura 1:** Logotipo de una fuente o canal web

---

Un canal web contiene una lista de las últimas actualizaciones, llamadas entradas. Normalmente, cada entrada tiene un título, un enlace para ampliar información, una descripción o resumen del contenido de la entrada y una fecha de publicación.

Por ejemplo, en la Figura 2 se muestra la página principal de RSS en el periódico El País (6). En este periódico, para cada sección se ha definido un canal diferente.

---

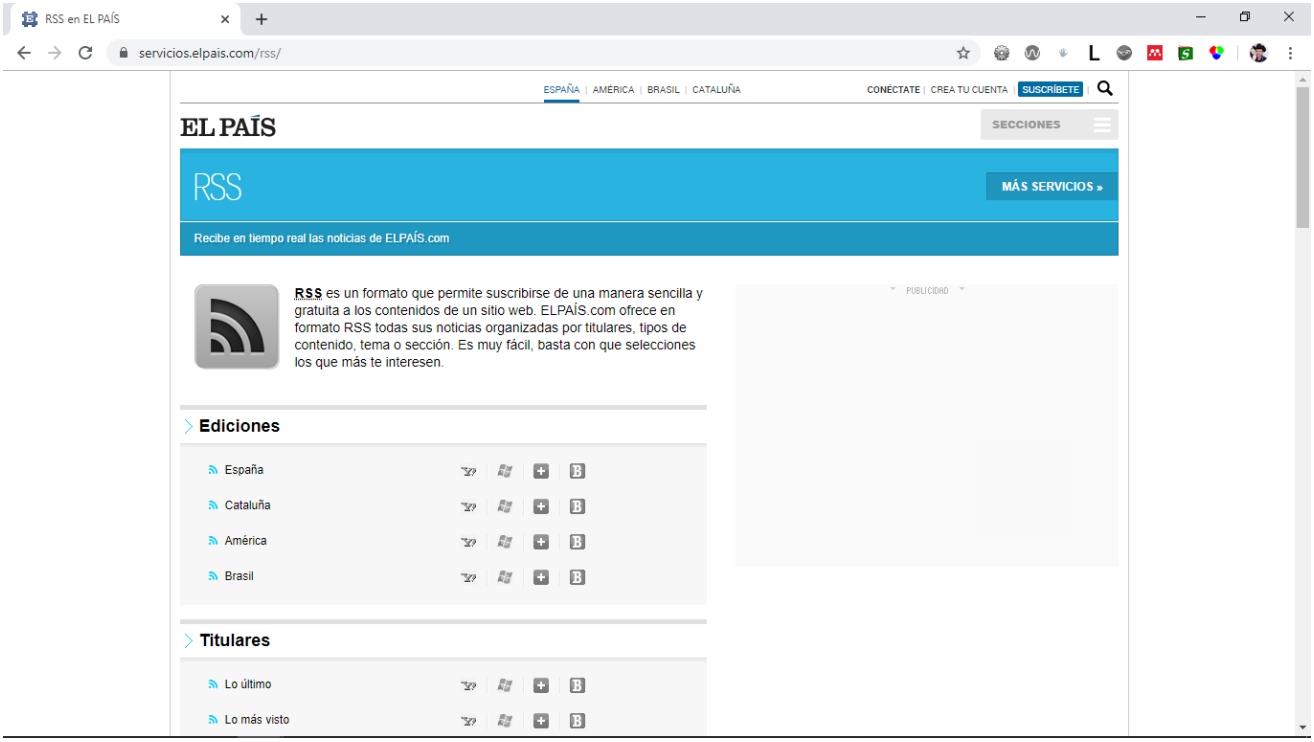


**Figura 2:** Página de canales en el periódico El País

Por ejemplo, en la Figura 3 se muestra el canal de la portada de la edición España del periódico (7). El navegador Google Chrome no es capaz de interpretar el código XML del documento, así que simplemente lo muestra.

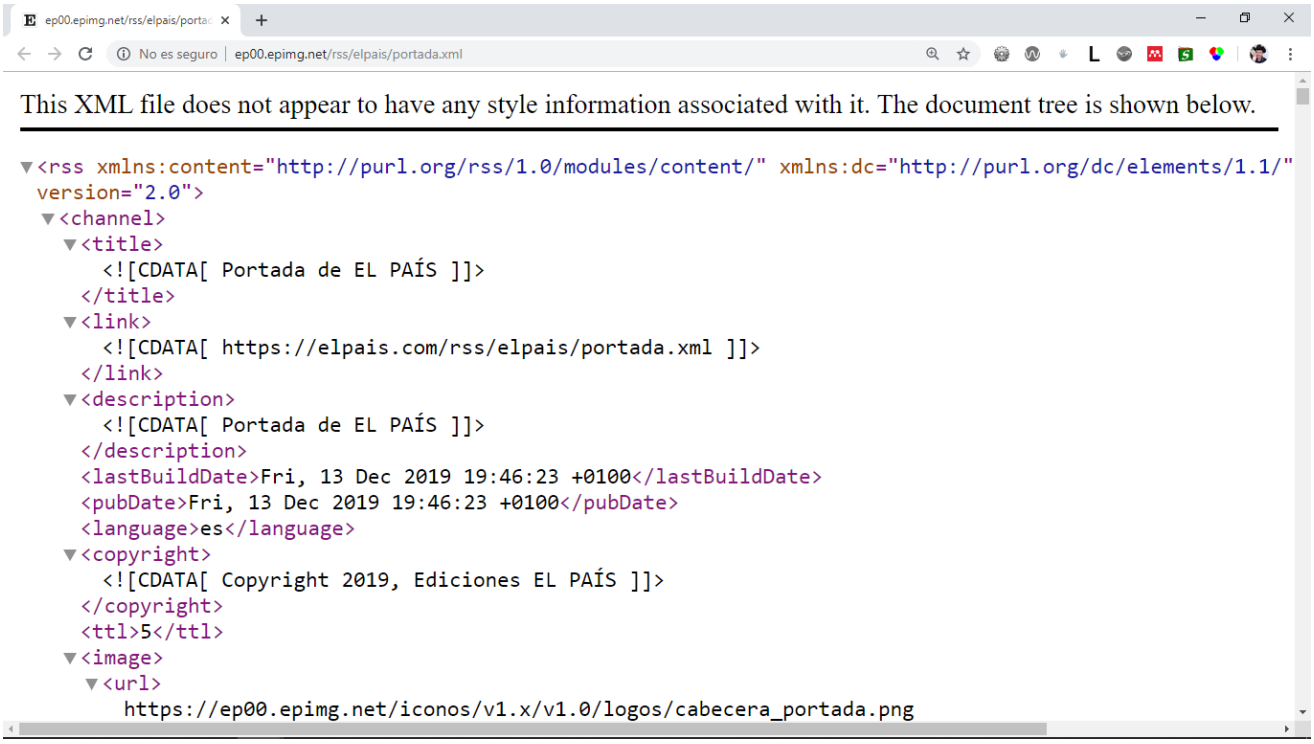


**Figura 3:** Canal de la portada del periódico El País

¿Cómo se puede visualizar un canal RSS? Hace falta utilizar un software especial, un lector de RSS. En el caso de Google Chrome es posible instalar una extensión, por ejemplo RSS Feed Reader (8) que se muestra en la Figura 4.

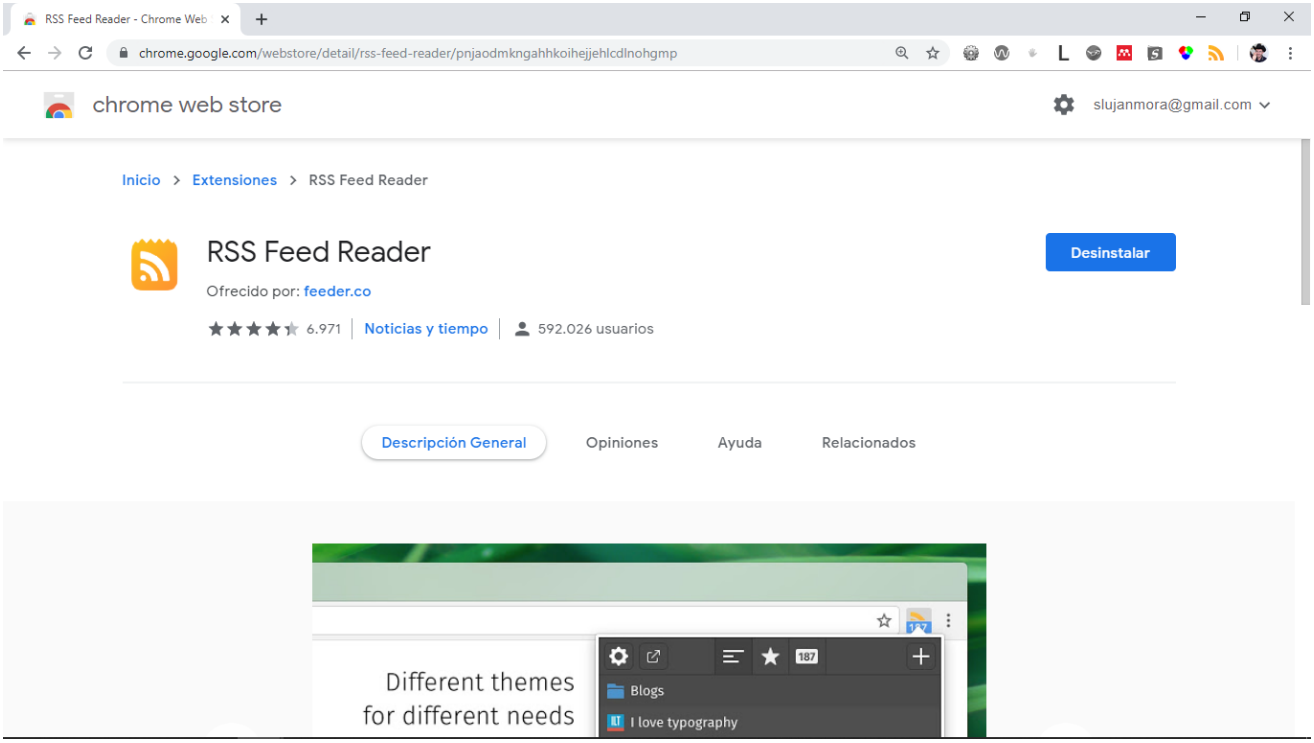


**Figura 4:** RSS Feed Reader

En la Figura 5 se muestra el mismo código XML que se visualiza en la Figura 3, pero interpretado a través de RSS Feed Reader.

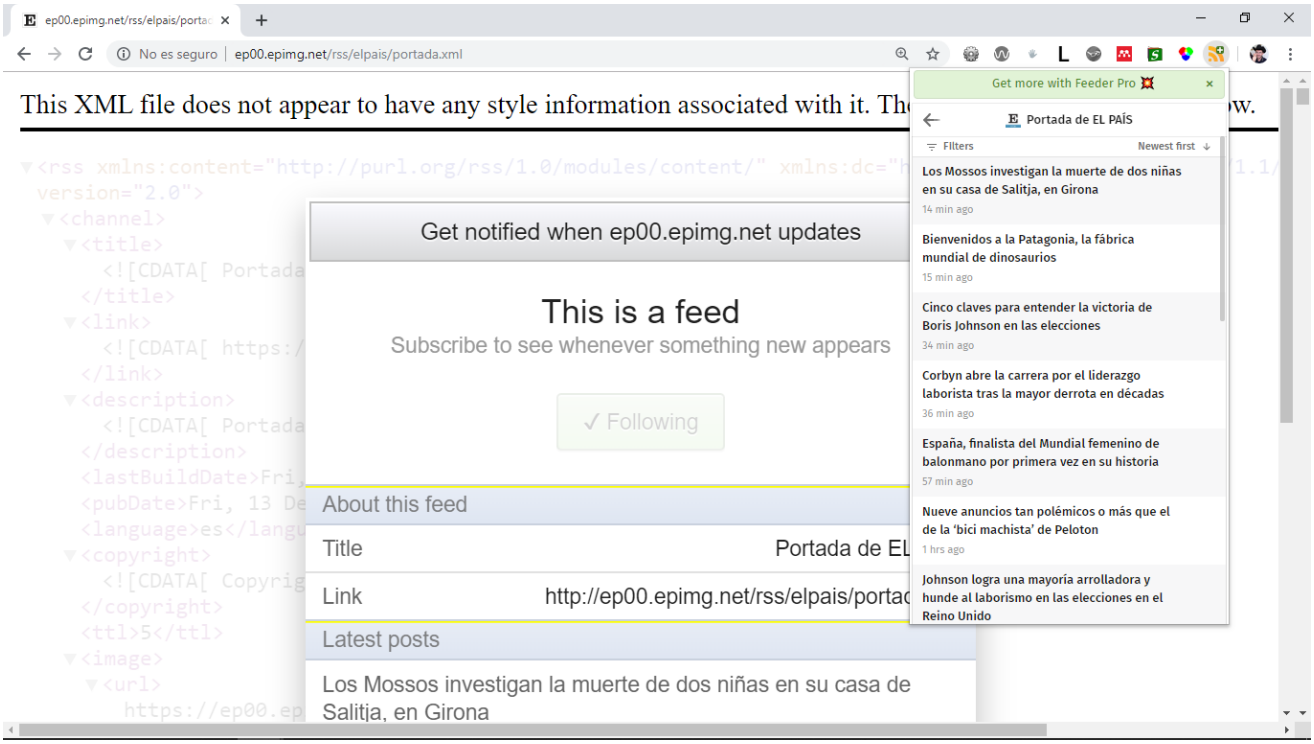

**Figura 5:** Canal de la portada del periódico El País visualizado a través de RSS Feed Reader

---

### 4.2. El DOM

El DOM (*Document Object Model*) es una API (*Application Programming Interface*) destinada a trabajar con documentos HTML y XML. El DOM está desarrollado por el W3C.

El DOM proporciona una representación en forma de árbol de un documento y permite su manipulación (añadir un elemento nuevo, borrar un elemento, mover un elemento de sitio).

PHP 5 posee la extensión DOM que permite manipular documentos HTML y XML. Esta extensión está instalada y configurada por defecto, por lo que en una instalación estándar no es necesario hacer nada especial para usarla.

La extensión DOM de PHP proporciona un conjunto de clases que representan diferentes partes de un documento HTML o XML. Las clases más importantes con algunos de sus métodos son:

- `DOMDocument`: representa un documento HTML o XML en su totalidad y sirve como raíz del árbol del documento.
  - `createAttribute()`: crea un nuevo atributo.
  - `createComment()`: crea un nuevo nodo de tipo comentario.
  - `createElement()`: crea un nuevo nodo de tipo elemento (etiqueta).
  - `createTextNode()`: crea un nuevo nodo de tipo texto.
  - `load()`: carga un documento XML desde un archivo.
  - `loadHTML()`: carga un documento HTML desde una cadena.
  - `loadHTMLFile()`: carga un documento HTML desde un archivo.
  - `loadXML()`: carga un documento XML desde una cadena.
  - `save()`: almacena el documento interno en un archivo usando el formato XML.
  - `saveHTML()`: devuelve el documento interno en forma de cadena usando el formato HTML.
  - `saveHTMLFile()`: almacena el documento interno en un archivo usando el formato HTML.
  - `saveXML()`: devuelve el documento interno en forma de cadena usando el formato XML.
  - `validate()`: valida un documento con su DTD.
- `DOMNode`: representa un nodo de un documento.
  - `appendChild()`: añade un nuevo hijo al final de los hijos.
  - `hasChildNodes()`: comprueba si el nodo tiene hijos.
  - `insertBefore()`: añade un nuevo hijo antes del nodo de referencia.
  - `removeChild()`: elimina un hijo de la lista de hijos del nodo.
  - `replaceChild()`: reemplaza un hijo.
- `DOMElement`: representa un elemento (etiqueta) de un documento. Hereda de la clase `DOMNode`.
  - `getAttribute()`: devuelve el valor de un atributo.
  - `setAttribute()`: añade un nuevo atributo.
  - `removeAttribute()`: elimina un atributo.
  - `hasAttribute()`: comprueba si existe un atributo concreto.

Un nodo (`DOMNode`) posee la propiedad `nodeType` que define el tipo de nodo. En el Cuadro 1 se muestran las constantes con sus respectivos valores para cada tipo de nodo.

---

| Constante | Valor | Tipo de nodo |
|---|---|---|
| XML_ELEMENT_NODE | 1 | DOMElement |
| XML_ATTRIBUTE_NODE | 2 | DOMAttr |
| XML_TEXT_NODE | 3 | DOMText |
| XML_CDATA_SECTION_NODE | 4 | DOMCharacterData |

| | | |
|---|---|---|
| XML_ENTITY_REF_NODE | 5 | DOMEntityReference |
| XML_ENTITY_NODE | 6 | DOMEntity |
| XML_PI_NODE | 7 | DOMProcessingInstruction |
| XML_COMMENT_NODE | 8 | DOMComment |
| XML_DOCUMENT_NODE | 9 | DOMDocument |
| XML_DOCUMENT_TYPE_NODE | 10 | DOMDocumentType |
| XML_DOCUMENT_FRAG_NODE | 11 | DOMDocumentFragment |
| XML_NOTATION_NODE | 12 | DOMNotation |

**Cuadro 1:** Constantes XML de tipo de nodo

Existen clases específicas para representar los diferentes tipos de nodos: `DOMComment`, `DOMElement`, `DOMText`, etc.

El siguiente ejemplo muestra cómo crear un documento HTML nodo a nodo. Para ello se proporciona la clase `HTMLDocument` que ofrece varios métodos para crear el contenido de una página web: `addHead()`, `addBody()`, `addStyleSheet()`, `addScript()`, `addMetaTag()`, etc. El método `generate()` genera la página web resultante en forma de cadena. La clase `HTMLDocument` permite generar páginas web con el DOCTYPE XHTML 1.0 Strict o HTML 5.

```
<?php
class HTMLDocument {
   private $doctype;
   private $lang;
   private $head;
   private $title;
   private $body;
   private $styles;
   private $metas;
   private $scripts;
   private $document;

   public function __construct($doctype = 'html5', $title = '', $lang = 'es') {
      $implementation = new DOMImplementation();

      switch($doctype) {
         default:
         case 'html5':
            $dtd = $implementation->createDocumentType('html');
            break;

         case 'xstrict':
            $dtd = $implementation->createDocumentType('html',
                     '-//W3C//DTD XHTML 1.0 Strict//EN',
                     'http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd');
            break;
      }

      $this->document = $implementation->createDocument('', '', $dtd);
      $this->lang = $lang;
      $this->doctype = $doctype;
      $this->title = $title;
   }

   public function addHead($element) {
      $this->head[] = $element;
   }

   public function addBody($element) {
      $this->body[] = $element;
```

```
    }

    public function addStyleSheet($url, $media = 'all') {
       $element = $this->document->createElement('link');
       $element->setAttribute('rel', 'stylesheet');
       $element->setAttribute('href', $url);
       $element->setAttribute('type', 'text/css');
       $element->setAttribute('media', $media);
       $this->styles[] = $element;
    }

    public function addScript($url) {
       $element = $this->document->createElement('script', '');
       $element->setAttribute('src', $url);
       $element->setAttribute('type', 'text/javascript');
       $this->scripts[] = $element;
    }

    public function addMetaTag($name, $content) {
       $element = $this->document->createElement('meta');
       $element->setAttribute('name', $name);
       $element->setAttribute('content', $content);
       $this->metas[] = $element;
    }

    public function setTitle($title) {
       $this->title = $title;
    }

    public function setAuthor($author) {
       $this->addMetaTag('author', $author);
    }

    public function setDescription($description) {
       $this->addMetaTag('description', $description);
    }

    public function setKeywords($keywords) {
       $this->addMetaTag('keywords', $keywords);
    }

    public function createElement ($nodeName, $nodeValue = NULL) {
       return $this->document->createElement($nodeName, $nodeValue);
    }

    public function generate() {
       $html = $this->document->createElement('html');
       $html->setAttribute('lang', $this->lang);
       switch($this->doctype) {
          case 'xstrict':
             $html->setAttribute('xmlns', 'http://www.w3.org/1999/xhtml');
             $html->setAttribute('xml:lang', $this->lang);
             break;
       }
       $this->document->appendChild($html);

       $head = $this->document->createElement('head', '');
       $title = $this->document->createElement('title', $this->title);
       $head->appendChild($title);

       if(is_array($this->metas)) {
          foreach($this->metas as $element) {
             $head->appendChild($element);
          }
       }
       if(is_array($this->styles)) {
          foreach($this->styles as $element) {
             $head->appendChild($element);
          }
```

```
        }
        if(is_array($this->scripts)) {
           foreach($this->scripts as $element) {
              $head->appendChild($element);
           }
        }
        if(is_array($this->head)) {
           foreach($this->head as $element) {
              $head->appendChild($element);
           }
        }
        $html->appendChild($head);

        $body = $this->document->createElement('body', '');
        if(is_array($this->body)) {
           foreach($this->body as $element) {
              $body->appendChild($element);
           }
        }
        $html->appendChild($body);

        switch($this->doctype) {
           default:
           case 'html5':
              $result = $this->document->saveHTML();
              break;

           case 'xstrict':
              $result = $this->document->saveXML();
              break;
        }

        return $result;
     }
  }

  // $document = new HTMLDocument("html5", "Esto es una prueba");
  $document = new HTMLDocument("xstrict", "Esto es una prueba");
  $document->addStyleSheet("estilo.css");
  $document->addScript("codigo.js");
  $document->setAuthor(utf8_encode("Sergio Luján Mora"));
  $document->setDescription(utf8_encode("Una página web de prueba"));

  $div1 = $document->createElement('div');
  $div1->nodeValue = 'Esto es un texto escrito en rojo.';
  $div1->setAttribute('style', 'color: red;');
  $document->addBody($div1);

  $div2 = $document->createElement('div');
  $div2->nodeValue = 'Esto es un texto escrito en verde.';
  $div2->setAttribute('style', 'color: green;');
  $document->addBody($div2);

  echo $document->generate();
  ?>
```

En el código anterior se ha usado la clase `DOMImplementation` para poder definir el DOCTYPE del documento que se genera, ya que la clase `DOMDocument` no ofrece un mecanismo para definirlo.

**¡Mucho cuidado!** La extensión DOM de PHP utiliza internamente la codificación UTF-8. Si la página PHP utiliza otra codificación, se tiene que hacer una conversión en la codificación de los caracteres. En el ejemplo anterior, como el código está escrito con la codificación ISO-8859-1, se ha usado la función `utf8_encode()` para realizar la conversión en aquellos casos en los que ha sido necesario (por ejemplo, cuando aparece una vocal acentuada).

**¡Mucho cuidado!** Un nodo sólo puede añadirse al documento a partir del cual ha sido creado, si se añade a otro documento se produce un error. Por ejemplo, en el siguiente código se crean dos nodos a partir de dos documentos y posteriormente se añaden al documento opuesto:

```
<?php
	$d1 = new DOMDocument();
	$d2 = new DOMDocument();

	$e1 = $d1->createElement("aaa");
	$d1->appendChild($e1);

	$e2 = $d2->createElement("bbb");
	$d2->appendChild($e2);

	$d1->appendChild($e2);
	$d2->appendChild($e1);
?>
```

El ejemplo anterior produce el siguiente mensaje de error:

```
Fatal error: Uncaught exception 'DOMException' with message 'Wrong Document Error' in
C:\Users\Sergio\Dropbox\htdocs\practicas\prueba.php:11
Stack trace:
#0 C:\Users\Sergio\Dropbox\htdocs\practicas\prueba.php(11):
DOMNode->appendChild(Object(DOMElement))
#1 {main}
 thrown in C:\Users\Sergio\Dropbox\htdocs\practicas\prueba.php on line 11
```

Si se quiere copiar un nodo de un documento a otro, se tiene que clonar (método `cloneNode()` de `DOMNode`) e importar (método `importNode()` de `DOMDocument`).

## 5. Recomendaciones

Crear un documento XML es muy sencillo con el DOM si se tiene clara la estructura del documento que se quiere crear. El primer paso antes de programar es construir el documento con “lápiz y papel”.

Puedes usar el validador Feed Validation Service (9) del W3C para comprobar que el formato del canal web es correcto.

---

(1) https://www.w3.org/TR/xml/

(2) https://web.ua.es/es/ice/pensemonline/mooc/tecnologias-de-intercambio-integracion-de-datos.html

(3) http://www.w3.org/DOM/

(4) http://php.net/manual/es/book.dom.php

(5) http://validator.w3.org/feed/

(6) https://servicios.elpais.com/rss/

(7) http://ep00.epimg.net/rss/elpais/portada.xml

(8) https://chrome.google.com/webstore/detail/rss-feed-reader/pnjaodmkngahhkoihejjehlcdlnohgmp

(9) http://validator.w3.org/feed/