Modelando entidades personalizadas con entradas en WordPress 4.0.1

Felix Ivan Romero Rodríguez, Codechanic, firomerorom4@gmail.com, La Habana, Cuba.

RESUMEN:

WordPress es un sistema gestor de contenido, conocido también como CMS, utilizado para la creación de blogs, portales, sitios promocionales. Aunque está diseñado con los fines antes expuestos es también utilizado para automatizar procesos de entidades y organizaciones lo que conlleve que se utilice como herramienta para la construcción de aplicaciones de gestión. En las aplicaciones de gestión se modelan procesos para listar, crear, editar y eliminar entidades. Para lograr ello usualmente se cambia la estructura del esquema original que propone WordPress en su mecanismo de persistencia. El presente artículo tiene como objetivo mostrar cómo gestionar entidades propias de procesos característicos de un entorno sin modificar el esquema original que propone WordPress para su base de datos. Para lograrlo se propone uso de patrones de diseño para el manejo de las entidades y componentes PHP para la carga y gestión de dependencias.

INTRODUCCION

WordPress (Patel, Rathod, & Parikh, 2011)es un sistema gestor de contenidos(Garcia, Xavier Cuerda, 2004) utilizado para la construcción de blogs, portales y webs promocionales. Aunque generalmente tiene los destinos antes mencionados, suele también emplearse para el desarrollo de procesos complejos como los procesos de gestión. En vista a ello se utiliza WordPress como herramienta de desarrollo utilizando los mecanismos que posee para listar, crear, editar y eliminar las entidades según el proceso o los procesos que se esté informatizando. Llevar a cabo la construcción de este tipo de sistema con WordPress, es realizado si bien modificando el modelo de datos original que propone WordPress o utilizando las herramientas que este CMS propone para trabajar sobre el suyo. Utilizando la primera vía conlleva a la modificación del estándar que utiliza WordPress lo que puede traer dificultades durante la actualización o migración del sistema. De la segunda manera sería utilizar herramientas como lo son las entradas o posts y sus datos asociados o postmetas para la gestión de los datos, teniendo en cuenta para ello el tipo de entrada, y los datos que implica cada uno de ellos. Mantener en cuenta todo ello se vuelve un proceso

engorroso cuando las entidades que se gestionan se incrementan de la misma manera que se complejizan los procesos en que intervienen. A raíz de esto la presente investigación tiene como objetivo proponer un mecanismo para facilitar la gestión de entidades en procesos informatizados con WordPress.

MATERIALES Y MÉTODOS

WordPress posee varios conceptos que son reflejados en el modelo de datos que ponemos. Contextualmente están las entradas o *posts*, los datos asociados o *postmetas* (Pallarés, 2015).

Las entradas constituyen el concepto más importante de WordPress, es el recurso que se maneja tanto para blogs, presentación o páginas comunes. A través de dichas tablas, este CMS, refleja los procesos de gestión que se realizan sobre las entradas.

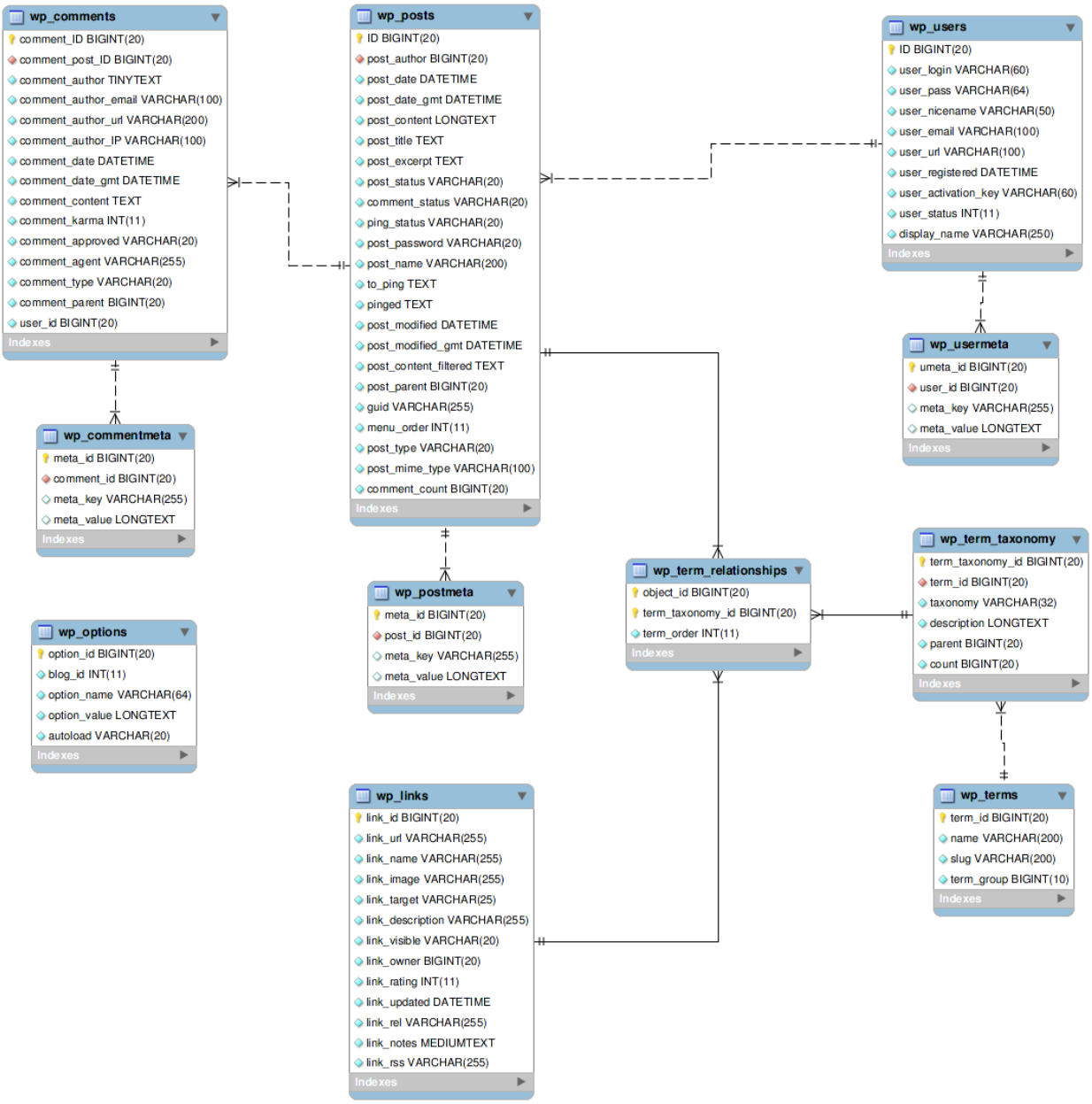

***Figura 1Modelo de datos de WordPress* (Pallis, 2010)**

Es válido aclarar la existencia de las entradas personalizadas. Utilizando las entradas personalizadas pueden agruparse según su tipo y realizarles un comportamiento personalizados cuando se les identifique. Es esta característica la utilizar la cual se usará como base para la gestión de entidades en procesos específicos.

Modelo Vista Controlador

El patrón arquitectónico modelo vista controlador, es utilizado para la construcción de aplicaciones web. Las capas de abstracción que propone, permite una

total interacción entre los niveles de la aplicación acentuando de igual manera la independencia entre estas. Este patrón logra establecer una fuerte separación entre la separación de las vistas, controladores y el modelo(Fowler, 2006) permitiendo que el cambio en una de las capas afecte de manera mínima cualquier componente en las otras. Esta facilidad de modificación brinda muchas facilidades para la construcción de aplicaciones web, permitiendo una independencia del diseño en ellas. Este patrón agrega flexibilidad en el diseño de sistemas web (LEFF, Avraham; RAYFIELD, 2001) lo que permite que sea utilizado en el diseño arquitectónico de marcos de trabajo y sistemas de gestión de contenidos.

Mediador

El patrón mediador, es un patrón estructural que permite la separación entre la aplicación y los objetos que se administran. Ello permite desacoplar los componentes de las aplicaciones y mediar entre la interacción de las entidades que se gestionan(Larman, 2011). Utilizar como práctica el mediador viabiliza la abstracción del manejo de objetos, de manera que el objeto mediador se utilice como el intermediario para acceder a los elementos almacenados.

RESULTADOS Y DISCUSIÓN

Primeramente, para aplicar la gestión de entidades en WordPress es necesario identificar dichas entidades y las operaciones que se realicen sobre ellas. Generalmente las operaciones responden a listar, crear, actualizar y eliminar entidades. Entonces para ello se crea la entidad con sus datos y el mediador que la gestiona.

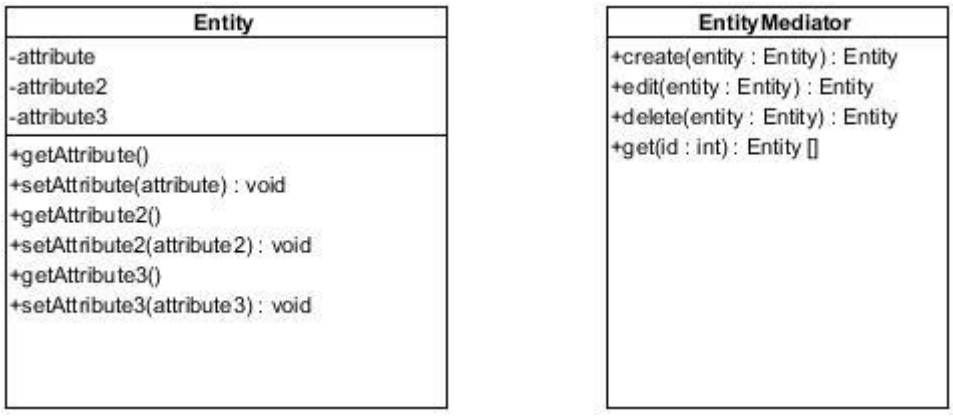

Figura 2 Entidad y su Mediador

Una vez identificado es entonces crear en el mediador la manera de comunicarse con WordPress para transformar las entidades en entradas. Esta transformación ocurre en los métodos de crear y editar, donde se debe crear una entrada cuyo

tipo sea una cadena que la relacione con la entidad y luego los atributos serían datos asociados o *postmetas* de dicha entrada. Una vez guardada la entrada con el método de WordPress *wp_insert_post()* se procede a guardar cada atributo como *postmeta* de dicha entrada o post. Una buena práctica sería implementar un método que devuelva en la entidad sus atributos en una colección de clave-valor, de manera que guardarlo como postmetas sería un proceso eficiente.

```php
class EntityManager {

    /**
     * @param Entity $entity
     * @return Entity
     * @throws \Exception
     */
    public function save(Entity $entity){

        try{

            $postID = wp_insert_post(
                array(
                    'post_title' =>$entity->getTitle() ,
                    'comment_status' => 'closed',
                    'post_status' => 'publish',
                    'post_type' => 'entity'
                ) );

            if ($postID)
            {
                $entity->setId($postID);

                $postmetas = $entity->toArray();
                foreach ($postmetas as $metakey => $metavalue) {

                        update_post_meta($postID,$metakey,$metavalue);

                }


                return $entity;
            }
            else{

                if ($postID instanceof \WP_Error) {
                    throw new \Exception($postID->get_error_message());
                }
```

Figura 3 Guardando información sobre la entidad

El proceso de actualizar se haría de manera similar, lo que en este caso se utilizaría la función *wp_update_post()*. Para recuperar los datos de la base de datos a través del mediador se utilizaría con las funciones *get_post()* y *get_postmeta()* de WordPress respectivamente. Una vez recuperados los datos, sería crear una colección de objetos de la entidad que se administra y devolverlas a través del método que las recupera de la base de datos.

```
     * @param $postID
     * @return array
     * @throws \Exception
     */
    public function get($postID){

        if (is_array($postID) && count($postID)>0) {

            return array_map(function($item){
                return $this->populate($item);
            },$postID);
        }
        else{

            return array(
                $this->populate($postID)
            );
        }
    }

    /**
     * @param $postID
     * @return Entity
     * @throws \Exception
     */
    protected function populate($postID){
        try{
            $entity = new Entity();
            $post = get_post($postID);
            $postmetas = get_post_meta($postID);
            $entity->setId($postID)
                    ->setTitle($post->post_title)
                    ->setStartDate($postmetas['_start_date'])
                    ->setEndDate($postmetas['_end_date'])

            return $entity;


        }
        catch(\Exception $e){
            throw $e;
        }
    }
```

Figura 4 Obteniendo datos

Para utilizar este mecanismo ello inicialmente se instancia un objeto de la entidad y se le asignan los datos. Luego se instancia la clase que se utiliza como mediador y se le pasa por parámetros el objeto con los datos y automáticamente. Lo siguiente sería incluir el sistema de entidades y mediadores dentro de WordPress. Es relevante saber que la lógica representada pertenece a la capa del modelo, según el patrón arquitectónico modelo vista controlador. Los procesos en WordPress pueden modelarse en plugins o en los temas. Una recomendable estructura sería colocar el proyecto con las clases de entidades y media-

dores dentro del directorio del tema y crear un archivo para auto cargar las clases. De esta manera en los ficheros donde se utiliza las entidades se incluye o requiere el archivo de auto carga y ya son accesibles las clases implementadas.

CONCLUSIONES

Modelar amplios procesos con WordPress de manera puede ser engorroso a medida que dichos procesos se complejizan. El uso de buenas prácticas como los patrones arquitectónicos y de diseño añade flexibilidad, escalabilidad y robustez, haciendo que el código sea reutilizable cuando se requiera mantener o añadir características a lo que se construye. Mediante el presente artículo se logra implementar un mecanismo que adapta las entidades de los procesos de un contexto en específico a los conceptos que trae consigo WordPress. Este mecanismo, al ajustarse al estándar de dicho CMS y utilizar patrones de diseño como el Mediador permite con esta modelación aprovechar las ventajas de la Programación Orientada a Objetos, así como una mayor precisión para manejar entidades salvadas como entradas y facilidad de extensión de las funcionalidades permitiendo un mejor diseño de los procesos que se informaticen.

REFERENCIAS

Fowler, M. (2006). GUI Architectures. Retrieved November 4, 2015, from http://martinfowler.com/eaaDev/uiArchs.html#ModelViewController

Garcia, Xavier Cuerda, and J. M. A. (2004). Introducción a los Sistemas de Gestión de Contenidos (CMS) de código abierto.

Larman, C. (2011). Appliying UML and Pattern. 8th Conference on the Advances of Computer Entertainment Technology. Retrieved from http://www.ace2011.org

LEFF, Avraham; RAYFIELD, J. T. (2001). Web-application development using the model/view/controller design pattern. In IEEE (Ed.), Enterprise Distributed Object Computing Conference, 2001. EDOC'01. Proceedings. Fifth IEEE International. IEEE (pp. 118–127). Retrieved from http://professor.unisinos.br/wp/crespo/files/2011/07/00950428.pdf

Pallarés, I. (2015). Desarrollo Web » Modelo de datos en Wordpress. Retrieved November 3, 2015, from http://www.ipallares.com/modelo-de-datos/

Pallis, G. (2010). Cloud Computing. IEEE Computer Society, 4. Retrieved from http://cgi.di.uoa.gr/~ad/MDE556/Papers/palis-ic10.pdf

Patel, S. K., Rathod, V. R., & Parikh, S. (2011). Joomla, Drupal and WordPress - a statistical comparison of open source CMS. 3rd International Conference on

Trendz in Information Sciences & Computing (TISC2011), 182–187. http://doi.org/10.1109/TISC.2011.6169111

Sociedad de la Información
Revista digital
D.L.: AB 293-2001
ISSN: 1578-326x

www.sociedadelainformacion.com

Edita:

Director: José Ángel Ruiz Felipe
Jefe de publicaciones: Antero Soria Luján
D.L.: AB 293-2001
ISSN: 1578-326x