12

Pocket Edition 2015

the original Hacker

creado por EUGENIA BAHIT

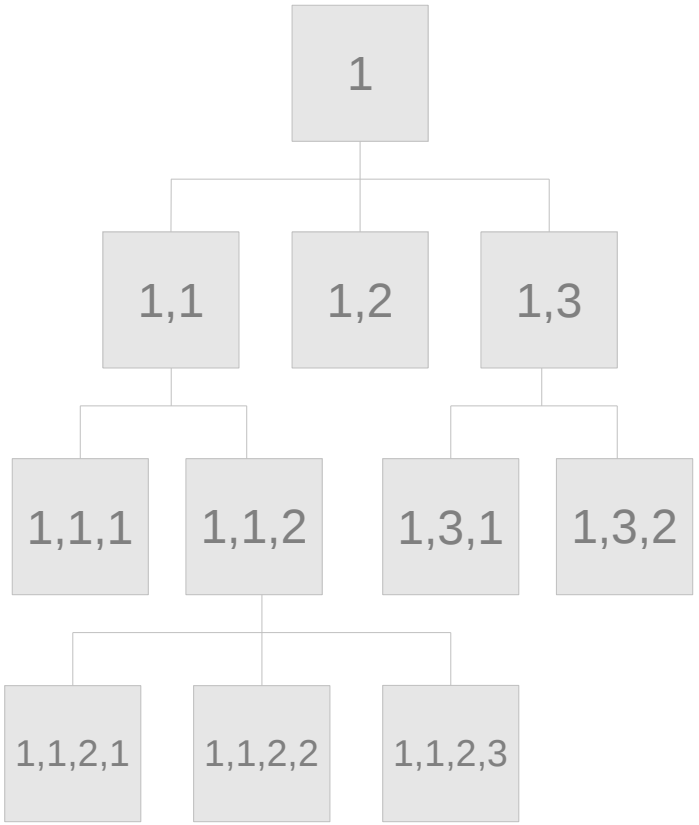
jugando con la inteligencia

OBJETOS RECURSIVOS

Categorías anidadas en Europio Engine

PROBLEMA

“El cliente necesita tener la libertad de anidar categorías unas dentro de otras con niveles de agrupamiento variable”



> *Lo que en realidad necesita el cliente se denomina «objeto recursivo»*

Objeto recursivo

Objeto que se compone de una colección de objetos de su mismo tipo

Objeto recursivo

La relación entre objetos recursivos es creada por un conector lógico

> *Un objeto recursivo se compone de sí mismo en forma cíclica, tantas veces como sea necesario*

7 Crear un modelo
heredado de BranchedObject

```
class Categoria extends BranchedObject {}
```

Si el modelo requiere de propiedades adicionales, agregarlas sobrescribiendo el método constructor:

```
class Categoria extends BranchedObject {

    public function __construct() {
        parent::__construct();
        $this->producto_collection = array();
    }
}
```

2 Crear la tabla
para persistencia del nuevo modelo

```
CREATE TABLE IF NOT EXISTS categoria (
    categoria_id INT(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
    denomination VARCHAR(60) NOT NULL,
    state INT(1) DEFAULT 0
) ENGINE=InnoDB;

/* crea una categoría raíz por defecto */
INSERT IGNORE
  INTO categoria (categoria_id, denomination, state)
VALUES (1, 'Root', 1);
```

denomination y **state** son dos propiedades heradadas que deben incluirse de forma obligatoria al mapear el modelo

3 Crear la tabla del conector lógico desde el CLI de Europio Engine

```
~$ ./europio -t
Para crear una nueva tabla, indique a qué tipo de objeto pertenece.
    (m)    Relacional Multiplicador
    (c)    Conector Lógico Relacional
    (s)    Objeto Serializado
    (q)    CANCELAR

Su opción (m/c/s/q): c
Nombre del Objeto Compuesto: Categoria
Nombre del Objeto Compositor: Categoria
Nombre de la base de datos: midatabase
QUERY:
================================================================
CREATE TABLE IF NOT EXISTS categoriacategoria ( connector_id INT(11) NOT NULL AUTO_INCREMENT
PRIMARY KEY , compuesto INT(11) NOT NULL , INDEX (compuesto) , FOREIGN KEY (compuesto)
REFERENCES categoria (categoria_id) ON DELETE CASCADE , compositor INT(11) NOT NULL , INDEX
(compositor) , FOREIGN KEY (compositor) REFERENCES categoria (categoria_id) ON DELETE
CASCADE )ENGINE=InnoDB;
Enter password:
Listo!
¿Desea crear otra tabla en foo? (s/n) n
Hasta luego!
~$
```

El CLI arrojará en pantalla la sentencia SQL que utilizará para crear la tabla

4 Crear formulario HTML
para agregar nuevas categorías

```
function agregar() {
    Dict::set_dict_for_webform($GLOBALS['plain_tree'], 'denomination');
    $form = new WebFormPRO('/modulo/categoria/guardar');
    $form->add_text('denomination', 'Nueva categoría:', null, null, 3);
    $form->add_select('categoria', 'Anidar en:', $GLOBALS['plain_tree'], null, 3);
    $form->add_hidden('state', 1);
    $form->add_submit('Guardar');
    $form->add_errorzone(array());
    $form->get_form();
    print Template('Agregar categoría')->show($form->form);
}
```

`$GLOBALS['plain_tree']` es un árbol de objetos anidados en texto plano, provisto por `BranchedObject`.

5 Completar el recurso “agregar” en el controlador para mostrar el formulario del paso 4

```
public function agregar() {
    $this->model->get_tree();
    $this->view->agregar();
}
```

`get_tree()` es el método encargado de agregar el árbol de objetos anidados al array $GLOBALS.

6 Completar el recurso “guardar” en el controlador para almacenar la nueva categoría

```
public function guardar() {
    extract($_POST);
    $this->model->denomination = $denomination;
    $this->model->state = $state;
    $this->model->save();
    $obj = Pattern::factory('Categoria', $categoria);
    $obj->add_categoria($this->model);
    $lc = new LogicalConnector($obj, 'Categoria');
    $lc->save();
    HTTPHelper::go("/modulo/categoria/ver");
}
```

7 Completar el método “ver” en la vista para mostrar el árbol completo de categorías

```
public function ver() {
    print "<pre>{$GLOBALS['plain_tree']}</pre>";
}
```

Esta vista puede ser mejorada
Tenga en cuenta que ésta, se trata de una vista temporal, solo a fines prácticos para entender el concepto de objeto recursivo.

8 Completar el recurso “ver” en el controlador

```
public function ver() {
    $this->model->get_tree();
    $this->view->ver();
}
```

Para probar, ingresar en:
http://<hostname>/<modulo>/categoria/agregar

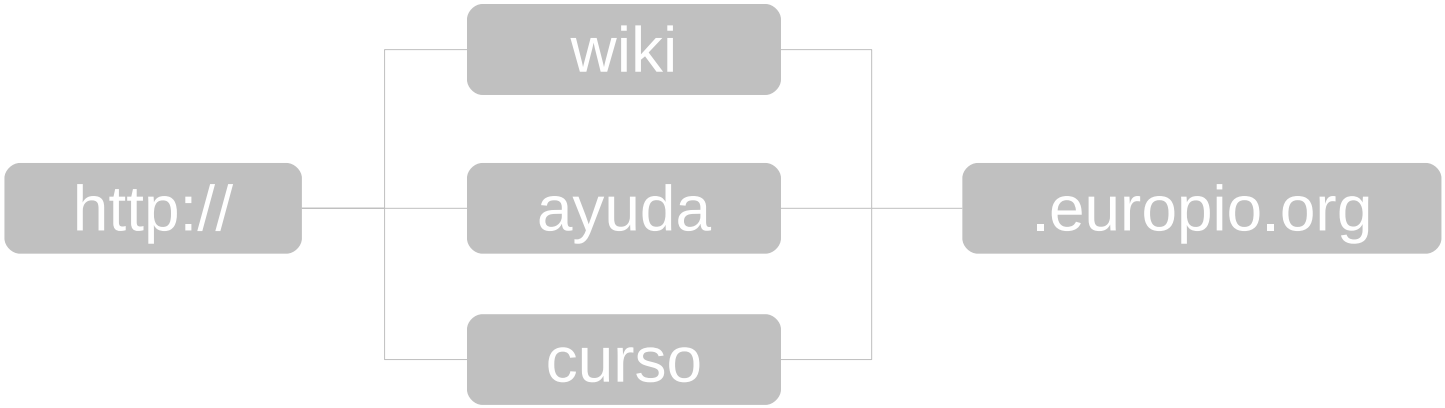
¿Te gustaría aprender más sobre Europio Engine?

Sitio Web oficial:
http://www.europio.org

Wiki del rpoyecto:
http://wiki.europio.org

Ayuda y soporte gratuito:
http://ayuda.europio.org

Curso online (profesional):
http://curso.europio.org



Eugenia Bahit

GLAMP Hacker & programadora eXtrema

Especializada en seguridad informática y desarrollo de Software mediante Ingeniería Inversa de código en Python y PHP.

Miembro de **Free Software Foundation** presidida por *Richard Stallman* (creador del proyecto GNU y el Software Libre) y **The Linux Foundation** presidida por *Linus Torvalds* (creador del kernel Linux).

Creadora de los proyectos **Europio Engine**, **Jack The Stripper**, **python-printr** y **Enhancement CLI for PHP**.

Fundadora de las revistas **The Original Hacker** y **Hackers** & **Developers Magazine**.

Dedicada actualmente a la docencia e investigación y a la **Formación Profesional de Hackers y Programadores**

WEB PERSONAL:
www.eugeniabahit.com

TWITTER:
@eugeniabahit

REPOSITORIOS:
http://repo.eugeniabahit.com

Licencia Libre

Creative Commons Atribución CompartirIgual

Eres libre de:

Compartir

copiar y distribuir este documento por cualquier medio y formato

Modificar

crear documentos derivados y distribuirlos

Pero:

Hazlo con respeto hacia al autor

Manteniendo la leyenda de copyright que ves al pie de cada página
(cuando hagas cambios, indica que eres el responsable de esos cambios)

No quites la libertad a otros

Cuando distribuyas este documento y/o tus modificaciones, deberás mantener la licencia original