Yatta

Martín Marrese <mmarre@mecon.gov.ar>

MEcon

Índice

1. Agregar un Proceso

El método YATTA_Controlador::agregarProceso es el encargado de dar de alta los nuevos procesos, y de ponerlos en la cola de ejecución. Este método recibe como parámetro una conexión a la base de datos y un array asociativo con los atributos del proceso. Devuelve un PEAR_Error en caso de haberlo.

1.1. Formato del array de datos

El array asociativo de datos está compuesto por claves obligatorias y claves opcionales. Es de suma importancia que se respete el formato de la clave (mayúsculas/minúsculas).

- Obligatorias

1. script : path completo del script. Con esto se hará el require once al momento de la ejecución.
2. id_sistema : es el identificador numérico asignado al sistema en SAMURAI.
3. descripción : es un texto descriptivo del proceso que será mostrado en la interfaz web al usuario.
4. owner : login@organismo del usuario que está iniciando el proceso. Si el proceso genera un archivo resultado, éste ocupará espacio en la cuenta de este usuario.

- Opcionales

1. destinos : login@organismo de los destinos. Son aquellos que tendrán acceso al resultado del proceso.
2. prioridad : indica la prioridad del proceso. Se utilizará para ponderar la ejecución de los sistemas.
3. notificar : indica si hay que avisarle al owner (y destinos) sobre la finalización de la ejecución del proceso.
4. resultado : este es el nombre del archivo resultado que hay que mostrarle al usuario en la interfaz web. No es necesariamente el mismo nombre que tendrá el archivo real de resultado.
5. parametros : este son aquellos parámetros que hay que pasarle al proceso al momento de la ejecución. Puede ser cualquier tipo de dato que pueda ser serializado y des-serializado correctamente.

2. Formato del Archivo Proceso

Cuando se quiere lanzar el proceso el script que corre en el servidor llama a una función que devuelve una instancia del objeto. Este tiene que cumplir ciertas condiciones para que pueda ejecutarse y obtener los datos cuando finaliza.

La función debe llamarse create_process, la cual no recibe parámetro alguno. Esto permite que el nombre y la ubicación de la clase que define al proceso sea transparente para el script.

El objeto debe tener un método run, que puede recibir parámetros o no, según se haya definido al cargarlo en la base de datos. Este método es invocado a la hora de ejecutar el proceso. Una vez que éste finaliza debe dejar en 3 atributos datos que son utilizados para dar por finalizados la ejecución.

- error : contiene cualquier mensaje de error que se hubiera producido durante la ejecución del proceso. Este será mostrado al usuario en la interfaz gráfica.

- archivo : contiene el path completo del archivo que se generó como resultado. Recomendación: utilizar nombres aleatorios para que el mismo script pueda ejecutarse repetidas veces sin riesgo a pisar los resultados. El usuario va a recibir como nombre el campo resultado de la base de datos que se cargo cuando se dio de alta el proceso.
- notificar : indica si hay que notificar al responsable y a los destinos o no, cambiando el valor seteado en la base.

3. Ejemplos

3.1. Alta Proceso

```
require_once 'YATTA/Controlador.php';
require_once 'DB.php';
require_once 'PEAR.php';
$db =& DB::connect('mysql://<user>:<passwd>@<host>/yatta', true);
if (DB::isError($db)) {
    trigger_error($db->getMessage(), E_USER_ERROR);
}
$datos = array (
    'script' => <path absoluto al archivo de create_process>,
    'id_sistema' => <id de samurai del sistema>,
    'descripcion' => <descripción que vera el usuario del proceso>,
    'owner' => <usuario@organismo owner>,
    'destinos' => <usuario@organismo destinos separados por ,>,
    'prioridad' => <ponderación de procesos, sin utilizar>,
    'notificar' => <bool>,
    'resultado' => <nombre del archivo que veran los usuarios>,
    'parametros' => <uno de string, int, array u objeto>
    );
$controlador =& new YATTA_Controlador;
$res = $controlador->agregarProceso($db, $datos);
if (PEAR::isError($res)) {
    trigger_error('Error: '. $res->getMessage() ."\n", E_USER_ERROR);\
}
```

3.2. Archivo Proceso

```
require_once 'YATTA/Proceso.php';
function create_process() {
    return new Mi_Proceso;
}

class Mi_Proceso extends YATTA_Proceso {
    function run($param) {
        //PUEDE HABER O NO PROCESOS
        //EJECUTO EL PROCESO
        ...
        $this->error = null; //si no hay error
        $this->notificar = false; //contradigo lo que dice en la base
        $this->archivo = /tmp/000res.zip; //archivo que muevo al tacho
    }
}
```