Example code Tuulia Antonius

Below, there are 2 examples from my resent research project with Symfony to make a server which receives log statements from other server through a REST API. It also has a web interface to read and search sent messages. These examples are shortened versions.

Table of Contents

Example controller

```php
<?php

        /**
        * This a Symfony controller class and is called when a client sends an API request to add a new
        * log to the database. The core function is insertLogAction. It has the following parts
        * 1) Read request databa
        * 2) Authorize
        * 3) Isert log an set satus code to OK. (If it is not OK an error is set in the inner classes)
        */
        namespace AppBundle\Controller;

        use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;
        use Symfony\Bundle\FrameworkBundle\Controller\Controller;
        use Symfony\Component\HttpFoundation\Request;
        use Symfony\Component\HttpFoundation\Response;
        use AppBundle\Manager\HttpExceptionManager;

        class ApiController extends Controller
        {


        /**
        * Insert New log
        *
        * ======================================================
        * Insert new log
        * ======================================================
        *
        * http POST "http:yourdomain//api/insertlog/" data:='__JSON__' --json -a
        * headers: "Authorization: Bearer API_KEY"
        *
        * Authorization API_KEY = Api key of the system
        * in the table `client_systems`
        *
        * __JSON__ = {"logData":{"type":"1","level":3,"message":"blaah222"}}
        *
        * __JSON__ in pretty format:
        * {
        * "logData": {
        * "type":"1",
        * "level":3,
        * "message":"blaah"
        * }
        * }
        *
        * ----------------------------
        * Curl example:
```

```
* ----------------------------
* With 'Test 3', who has 'apikey' API_KEY in the table `client_systems`
* >curl \
* --header "Authorization: Bearer API_KEY" \
* http://loggingserver.nl/api/insertlog/ \
* POST \
* -H "Content-Type: application/json" \
* -d '{"logData":{"type":"1","level":3,"message":"blaah222"}}' \
*
*
* Response:
* (id of the new log in table 'logs'
*
* [41]
*
*
* @Route("/api/insertlog/")
*/
public function insertLogAction()
{
$data = $this->getRequestData();
$systemId = $this->authorize($data);

$logManager = $this->get('AppBundle.LogsManager');
$logId = $logManager->insert($systemId, $data->logData, $data->xSignature );

$response = new Response(json_encode([$logId]));
$this->log(get_class($response));
$response->setStatusCode(HttpExceptionManager::$CREATED);
$response->headers->set('Content-Type', 'application/json');
return $response;
}


/**
* Authorize
*
* This should be called in every function to ensure authorize access.
*
* @param $data
*/
private function authorize($data) {
$systemManager = $this->get('AppBundle.ApiAuthorizationManager');
return $systemManager->authorize($data);

}
```

```
/**
* Get request data from post and headers of the request
*
* Separate some important params:
* data All request data
* apiKey Clear api key (as in client_systems.apikey)
*
*
* @return object
*/
private function getRequestData()
{
$request = Request::createFromGlobals();
$fileData = json_decode(file_get_contents('php://input'));
return (object)[
'data' => $_REQUEST,
'logData' => isset($fileData->logData) ? $fileData->logData : false,
'apiKey' => $request->headers->get('authorization'),
'xSignature' => isset($_GET['xSignature']) ? $_GET['xSignature'] : false,

];
}

/**
* Set log statement
*
* @param string $message
* @param string $type
*/
private function log($message , $type = 'debug') {
$this->get('appBundle.MonoLogManager')->log($message , $type);
}

}
```

Example class

```php
<?php
        /**
        * This a Symfony manager class as an abstract base class for all of the
        * business rules of database tables.
        * Symfony has an entity class for each table for the database transactions.
        * This is a manger class used as an extension for the Symfony entity classes.
        * This is a shortened version.

        * This class includes functions and properties,which must be included in every
        * entity class to manage the business rules.
        *
        */
        namespace AppBundle\Manager\Manager;

        use AppBundle\Manager\Manager;
        use Symfony\Component\HttpFoundation\Response;
        use Doctrine\ORM\Tools\Pagination\Paginator;

        abstract class EntityManager extends Manager {

        /**
        * Id field name of the current table

        * @var string
        */
        private $idFieldName;

        /**
        * Set entity manager parameters
        *
        * This a Symfony function. It is sort constructor, to initialize the object with
        * given values.
        *
        * @param Symfony\Component\Validator\Validator\RecursiveValidator $validator
        * @param string $idFieldName
        * @param string $repositoryName
        */
        public function setEntityManagerParams($validator, $idFieldName, $repositoryName) {

           $this->validator = $validator;
           $this->idFieldName = $idFieldName;
           $this->repositoryName = $repositoryName;
        }
```

```php
/**
* Find given field
*
* This is function executes a SQL query in format :
* SELECT * FROM $this->repositoryName where $field = $value
*
* As default this finds by the id of the table ($this->idFieldName).
*
* @param $value Value to search
* @param $field Database column to search
* @param array $options options
* 'toArray' If this variable is set:
* Convert all results from objects to array
* @return array
*
*/
public function find($value , $field = false, $options= []) {
  // If no search field defined, use the default one
  if (!$field) {
    $field = $this->idFieldName;
  }
  // make method name
  // Convert $this->fieldname xxx_yyy_zzz to findByXxxYyyZzz
  $methodName = ucwords(str_replace('_', ' ' , $field ));
  $methodName = 'findBy' . str_replace(' ', '' , $methodName);

  $res = $this->em
    ->getRepository($this->repositoryName)
    ->$methodName($value);

  if (isset($options['toArray'])) {
    $res = $this->toArray($res);
  }
return $res;
}



/**
* Update or insert to database
*
* Thus is a generic function to update and insert to database.
* You just need send as parameters an array of data to be updates
*
* @param array $data Data to be inserted or updated.
* Example, insert:
* [ name => 'John Doe']
* Example update:
```

```
* [ 'i=> 23, name => 'John Smith']
*
* @return integer Id of the item. If you insert you get the id of the
* new insert.
*/
public function save($data) {

  $this->log('Start function ' .__CLASS__ . '::' . __FUNCTION__ . ' with data: ' .
   print_r($data, true) );

   // If id exists this is update
   $action = isset($data[$this→idFieldName]) ? ‘UPDATE' : 'INSERT';


  // Define entity after the query type
   $entity = ($action == 'INSERT')
      ? $entity = $this->getDatabaseEntity()
      : current($this->find($data[$this->idFieldName]));

   // Set data
   array_walk(
     $data,
     function($value, $key, &$entity) {
       $entity->set($key, $value);
     },
     $entity
   );

  // Execute query
  $this->em->persist($entity);
  $this->em->flush();

  // return id
  return $entity->get($this->idFieldName);
}


/**
* Generic validate function
*
* This uses the Symfony validator class to validate.
* (The validation rules are defined in the table entity classes)
*
* @param array $data See function 'save'
* @return array Set of error messages
*/
public function validate($data) {
   $entity = $this->getDatabaseEntity();
   array_walk(
```

```
        $data,
        function($value, $key, &$entity) {
          $entity->set($key, $value);
        },
        $entity
    );

    $errors = $this->validator->validate($entity);
    $erroMessages = [];
    for($index = 0 ; $index < $errors->count(); $index++){
        $erroMessages[] = $errors->get($index)->getMessage();

    }

    return $erroMessages;
}

/**
* Search with given parameters
*
* This function is used on an index page of my project, where you can
* search by given criteria (filters) and order them by given criteria (sort).
* As a result you get a list of rows. If the paging is set, only a section of
* results on the current page is returned.
*
* This function creates dynamically a query with the given parameters.
*
* Example of filters:
*
* With this filter we filter:
* 'timestamp' from table 'AppBundle:Logs' ( = L )
* 'name' from table 'AppBundle:ClientSystems' ( = S )
*
* $filters = [
* [
* 'field' => 'L.timestamp',
* 'operator' => '>=',
* 'value' => '2016-07-20',
* ],
* [
* 'field' => 'S.name',
* 'operator' => '=',
* 'value' => 'Test 2',
* ],
*];
*
* Example of sort:
```

```
*
* $sort = [
* ['field' => 'L.type', 'dir' => 'ASC'],
* ['field' => 'L.level', 'dir' => 'DESC'],
* ];
*
* See examples how to use this in
* Tests\AppBundle\Controller\frontEndControllertest : testList() and testListPagination()
* and in class AppBundle\Manager\Manager\Entity\Logs : getSystemsList()
*
* @param Doctrine\ORM\QueryBuilder $baseQuery The base query
* @param array $filters See above
* @param array $sort See above
* @param integer $page If given the current page is returned, others the whole set
* without limits
* @param integer $maxItemsPerPage If not defined the configured variable 'list_items_per_page'
* is used.
* @return \stdClass This returns an object with following members
* 'list' => array of result records,
* 'totalPages' => total amount Pages,
* 'totalRecords' => total amount Records
*/
protected function getList($baseQuery, $filters = [], $sort = [], $page = 0, $maxItemsPerPage = false) {

   // Add filters
   array_walk(
     $filters,
     function($filter, $values, &$baseQuery) {
        $paramName = 'param_' . uniqid(); // unique id is to avoid delicate parameter names
        $where = $filter['field'] . ' ' . $filter['operator'] . ' :' . $paramName ;
        $baseQuery->andWhere($where)
          ->setParameter($paramName, $filter['value']);
        },
      $baseQuery
   );

   //Add order by
   array_walk(
      $sort,
      function($sort, $values, &$baseQuery) {
         if (!empty($sort)) {
           $baseQuery->addOrderBy($sort['field'] , $sort['dir']);
         }
      },
      $baseQuery
    );

   // Do query
```

```
    $query = $baseQuery->getQuery();

    // Get total number of items and pages
    $paginator = new Paginator($query, $fetchJoinCollection = false);
    $totalRecords = $paginator->count();
    $numberOfItemPerPage = ($maxItemsPerPage)
    ? $maxItemsPerPage : $this->getContainer()->getParameter('list_items_per_page');
    $totalPages = ceil($totalRecords / $numberOfItemPerPage );

    // Do pagination
    if ($page) {
    $page = $page - 1;
    $firstResult = $page * $numberOfItemPerPage;
    $query->setFirstResult($firstResult)->setMaxResults($numberOfItemPerPage);
   }

    //var_dump($query->getSql()); // debug query

    $res = $query->getResult();

    return (object)[
        'list' => $res,
        'totalPages' => $totalPages,
        'totalRecords' => $totalRecords
        ];
    }

/**
* Get menu
*
* This menu can be used with Symfony From class and method:
* AppBundle\Manager\Manager\Form::get()
*
* @param string $labelField
* @return array
*/
protected function getMenu($labelField) {

    $baseQuery = $this->em->createQueryBuilder()
        ->select('S')
        ->from($this->getRepositoryName(), 'S')
        ->orderBy('S.name');
    $query = $baseQuery->getQuery();
    $systems = $query->getResult();
    $menu = [];
    array_map(
        function ($system) use (&$menu, $labelField){
          $menu[] = [
```

```
            'name' => $system->get($labelField),
            'value' => $system->get($this->idFieldName)
           ];

        },
        $systems
    );
    return $menu;
}

}
```