QueryBuilder

There is a library just for turning $_GET requests into QueryBuilder parameters for Doctrine. This library was written about the same time as Doctrine in Apigility and it is intended to be used together.

You can find zfcampus/zf-doctrine-querybuilder here. Please read the Philosopy first to understand why this library was written and to ensure you're comfortable with the excellent level of access the library provides to clients of your API. The rest of the documentation for that repository details well the capabilities so here I'll show a complete example for implementing zf-doctrine-querybuilder with Query Providers.

Abstract Factory

We'll need a common factory for all Query Providers to implement zf-doctrine-querybuilder

```
<?php
  namespace DbApi\Query\Provider;

  use Interop\Container\ContainerInterface;
  use Zend\ServiceManager\Factory\AbstractFactoryInterface;
  use Doctrine\Instantiator\Instantiator;
  use ZF\Doctrine\QueryBuilder\Filter\Service\ORMFilterManager;
  use ZF\Doctrine\QueryBuilder\OrderBy\Service\ORMOrderByManager;

  class QueryProviderAbstractFactory implements
      AbstractFactoryInterface
  {
      public function canCreate(ContainerInterface $container, $requestedName)
      {
          $instantiator = new Instantiator();
          $instance = $instantiator->instantiate($requestedName);

          return ($instance instanceof AbstractQueryProvider);
      }

      public function __invoke(ContainerInterface $container, $requestedName, array
$options = null)
      {
          $instance = new $requestedName();
          $instance->setFilterManager($container->get(ORMFilterManager::class));
          $instance->setOrderByManager($container->get(ORMOrderByManager::class));
          $instance->setObjectManager($container-
>get('doctrine.entitymanager.orm_default'));
          $instance->setAuthentication($container->get('authentication'));

          return $instance;
      }
  }
```

We're including the filter and order by manager from zf-doctrine-querybuilder in our Query Providers. Additionally the Object Manager is set (note for this class the object manager alias is static) and the authentication from zf-mvc-auth is set. This gives access to the currently authenticated user.

Abstract Query Provider

Each Query Provider created through this abstract factory must be a AbstractQueryProvider and here is that code

```php
<?php
  namespace DbApi\Query\Provider;

  use Zend\Authentication\AuthenticationService;
  use ZF\Apigility\Doctrine\Server\Query\Provider\AbstractQueryProvider as
ZFCampusAbstractQueryProvider;
  use ZF\Doctrine\QueryBuilder\Filter\Service\ORMFilterManager;
  use ZF\Doctrine\QueryBuilder\OrderBy\Service\ORMOrderByManager;
  use ZF\Rest\ResourceEvent;
  use DoctrineModule\Persistence\ProvidesObjectManager;

  abstract class AbstractQueryProvider extends ZFCampusAbstractQueryProvider
  {
      use ProvidesObjectManager;

      private $filterManager;
      private $orderByManager;
      private $authentication;

      public function getAuthentication()
      {
          return $this->authentication;
      }

      public function setAuthentication(AuthenticationService $authentication)
      {
          $this->authentication = $authentication;
      }

      public function getFilterManager()
      {
          return $this->filterManager;
      }

      public function setFilterManager(ORMFilterManager $filterManager)
      {
          $this->filterManager = $filterManager;

          return $this;
      }

      public function getOrderByManager()
      {
          return $this->orderByManager;
      }

      public function setOrderByManager(ORMOrderByManager $orderByManager)
      {
          $this->orderByManager = $orderByManager;

          return $this;
      }

      public function createQuery(ResourceEvent $event, $entityClass, $parameters)
      {
          $request = $event->getRequest()->getQuery()->toArray();
          $queryBuilder = $this->getObjectManager()->createQueryBuilder();
          $queryBuilder->select('row')
              ->from($entityClass, 'row');

          if (isset($request['filter'])) {
              $metadata = $this->getObjectManager()->getClassMetadata($entityClass);
              $this->getFilterManager()->filter(
                  $queryBuilder,
                  $metadata,
                  $request['filter']
              );
```

```
        }

        if (isset($request['order-by'])) {
            $metadata = $this->getObjectManager()->getClassMetadata($entityClass);
            $this->getOrderByManager()->orderBy(
                $queryBuilder,
                $metadata,
                $request['order-by']
            );
        }

        return $queryBuilder;
    }
}
```

The interesting function here is `createQuery`. This function is part of the ZFCampusAbstractQueryProvider's interface. With this we parse the Request's query() data and send it through the filter manager and order by manager. These managers apply the filters from the query to the QueryBuilder.

Configuration

Enable the abstract factory for zf-apigility-doctrine-query-provider

```
<?php
  'zf-apigility-doctrine-query-provider' => array(
      'abstract_factories' => array(
          'DbApi\\Query\\Provider\\QueryProviderAbstractFactory',
      ),
  ),
```

Query Provider Example

To create a query provider extend it from the new AbstractQueryProvider and call the parent createQuery as the first line of the `createQuery` function

```php
<?php
  namespace DbApi\Query\Provider;

  use ZF\Rest\ResourceEvent;
  use DbApi\Query\Provider\AbstractQueryProvider;
  use Db\Fixture\RoleFixture;

  final class PerformanceCorrectionPatch extends AbstractQueryProvider
  {
      public function createQuery(ResourceEvent $event, $entityClass, $parameters)
      {
          $queryBuilder = parent::createQuery($event, $entityClass, $parameters);

          if ($this->getAuthentication()->getIdentity()->getUser()-
>hasRole(RoleFixture::$ADMIN)) {
              return $queryBuilder;
          }

          // The creating user can edit this
          $queryBuilder
              ->andWhere($queryBuilder->expr()->eq('row.user', ':user'))
              ->setParameter('user', $this->getAuthentication()->getIdentity()-
>getUser())
              ;

          return $queryBuilder;
      }
  }
```

Note