Introduction to EFQ

About me

Joshua Li

Drupal developer in DATACOM

d.o ID: rli

I also open the door and order pizza for you …


WARNING

What is EFQ ?

class EntityFieldQuery

"This class allows finding entities based on entity properties (for example, node->changed), field values, and generic entity meta data (bundle, entity type, entity id, and revision ID)." -- drupal.org

So, it is just another way to get node and other entities …

The ways we know to find a node or other entity

- db_query, db_select, db_.....
- Views module
- and …?
- EntityFieldQuery

So why EFQ??

- db_query/db_select
  - I don’t know SQL …
  - And I don’t understand “leftJoin”, “innerJoin”, whatever join
- The great Views module
  - Not in core yet, (soon, can’t wait), but still not.
  - Slowing down the website, complicated query.
  - Accessible by anyone with the right permission
- So maybe think about EFQ ??

What exactly does EFQ look like?

- Easier to explain when you see the code

```php
<?php
$query = new EntityFieldQuery();

$query->entityCondition('entity_type', 'node')
  ->entityCondition('bundle', 'article')
  ->propertyCondition('status', 1)
  ->fieldCondition('field_news_types', 'value', 'spotlight', '=')
  ->fieldCondition('field_photo', 'fid', 'NULL', '!=')
  ->fieldCondition('field_faculty_tag', 'tid', $value)
  ->fieldCondition('field_news_publishdate', 'value', $year. '%', 'like')
  ->fieldOrderBy('field_photo', 'fid', 'DESC')
  ->range(0, 10)
  ->addMetaData('account', user_load(1)); // Run the query as user 1.

$result = $query->execute();

if (isset($result['node'])) {
  $news_items_nids = array_keys($result['node']);
  $news_items = entity_load('node', $news_items_nids);
}
?>
```

Like the db_s but easier

Layer in front of db

- So we don’t care what’s the table name anymore
- no more multiple tables
- much more flexible (case study coming)

Extendable

- As it is a class, we can define our own class based on EFQ i.e list all published nodes

extendable class example

```
class NodeEntityFieldQuery extends EntityFieldQuery {
 // define some defaults for the class
 public function __construct() {
  // now we don't need to define these over and over anymore
  $this
   ->entityCondition('entity_type', 'node')
   ->propertyCondition('status', 1)
   ->propertyOrderBy('created', 'DESC');
  // define a pager
  $this->pager();
 }

 public function excludeNode($nid) {
  // code snip; we'll come back to this.
 }

}
```

Example from http://www.phase2technology.com/blog/entityfieldquery-let-drupal-do-the-heavy-lifting-pt-2/

extendable class example continue

```php
/**
 * If we're on a node, and if the entity_type is node, exclude the local node from the query
 */
public function excludeNode($nid) {
  if (!$nid) {
    $object = menu_get_object();
    $nid = $object->nid;
  }
  if (!empty($nid) && $this->entityConditions['entity_type']['value'] === 'node') {
    $this->propertyCondition('nid', $nid, '<>');
  }
  return $this;
}
```

Case study

Flippy module:

showing pagers for each content type, nice and easy, welcome to use.

You can guess, of course it was using db_select.

Everyone was happy with it until one day ...

Case study

People wanted to sort the nodes by different fields.

And I don’t which field and what field they have... And how many joins do I need to do with db_select???

Case study

And something more serious...

I got contacted by the drupal security team...

it was an honor …

Problem/Motivation

The Flippy module generates previous/next links for nodes of selected content types. However these links to do not take the node access rights of their targets into account: users may thus get a previous/next link that points to a node they do not have access to.

I like how they described it.

Case study

And you know the solution:

```
$query = new EntityFieldQuery();
$query->entityCondition('entity_type', 'node')
  ->entityCondition('bundle', $node->type)
  ->propertyCondition('status', 1)
  ->propertyCondition('nid', $node->nid, '!=')
  ->propertyCondition('language', array($language->language, LANGUAGE_NONE), 'IN')
  ->range(0, 1)
  ->addTag('node_access');
```

```
if ($field_value) {
  // set the conditions
  $first->fieldCondition($sort, 'value', $field_value, $before);
  $prev->fieldCondition($sort, 'value', $field_value, $before);
  $next->fieldCondition($sort, 'value', $field_value, $after);
  $last->fieldCondition($sort, 'value', $field_value, $after);
  // set the ordering
  $first->fieldOrderBy($sort, 'value', $up);
  $prev->fieldOrderBy($sort, 'value', $down);
  $next->fieldOrderBy($sort, 'value', $up);
  $last->fieldOrderBy($sort, 'value', $down);
}
```

Case study

addTag and hook_query_TAG_alter()

```
$random->addTag('random');
$result = $random->execute();
```

```
/**
 * Implement hook_query_TAG_alter()
 */
function flippy_query_random_alter($query){
    $query->orderRandom();
}
```

Still needs work

Doesn't support db_or/db_and.

Drupal 8

Coming soon... (???)

Work around:

addTag, and '$query->join', 'db_or' in hook_query_TAG_alter()

EFQ in D8

andConditionGroup/orConditionGroup

For example, consider a map entity with an 'attributes' field containing 'building_type' and 'color' columns. To find all green and red bikesheds:

```
$query = Drupal::entityQuery('map');
$group = $query->orConditionGroup()
  ->condition('attributes.color', 'red')
  ->condition('attributes.color', 'green');
$entity_ids = $query
  ->condition('attributes.building_type', 'bikeshed')
  ->condition($group)
  ->execute();
```

Userful links

- https://api.drupal.org/api/drupal/includes!entity.inc/class/EntityFieldQuery/7
- https://drupal.org/node/1343708
- http://www.phase2technology.com/blog/building-energy-gov-without-views/
- http://www.phase2technology.com/blog/or-queries-with-entityfieldquery/

Questions ??