Fields, Entities and Lists, Oh My!

October 24, 2015

Phase2

Saturday, October 24, 15

Good morning everyone, and welcome to Fields, Entities and Lists, Oh My!

Joshua Turton

Senior Developer

Email: jturton@phase2technology.com

Twitter: @sjinteractive and @phase2

Drupal.org: srjosh

Saturday, October 24, 15

[Introduction of me]

Today we'll be talking about some important concepts and tools that are key to developing in Drupal 7 - Fields, Entities, and Lists. We'll be talking about them mostly from a code perspective, so I'm going to make the assumption that you know at least enough to make a basic module and get Drupal to recognize and enable it.

The things I'm covering today are all items I use practically every week, and a good working knowledge of them will make your life as a developer much more productive.

Entities and Fields

Saturday, October 24, 15

We'll start with two of the most powerful and widely used tools in Drupal – entities and fields.

Introduced in a limited fashion in Drupal 5 & 6 with "Nodes", they became standardized in Drupal 7, and they are everywhere in Drupal 8. Knowledge of what they can do and how to use them is critical.

At their core, entities are units of information. They are PHP objects, defined by classes, which allow for the storage and manipulation of various types of data.

Core Entity Types

- Nodes
- Comments
- Taxonomy Terms
- User Profiles

These are the entity types that ship with Drupal core. Contributed modules greatly expand this list, and you can create your own entities as well!

Drupal also allows you to define different "types" of those entities, to use for different purposes.
[Discuss:
 * Show Content Types
 * Talk about usefulness via Fields
 * Show Fields admin for node
 * Show Fields admin for taxonomy type, talk about standardization of interface
]

Core Field Types (and their modules)

- Textfield (text)
- Textarea (text)
- Textarea with Summary (text)
- File (file)
- Image (image)
- Radio Buttons (options)
- Checkboxes (options)
- Single Select (options)
- Multi-select (options)
- Taxonomy Term (taxonomy)
- Decimal (number)
- Float (number)
- Integer (number)

These are the field types that ship with Drupal core. Contributed modules greatly expand this list, and you can create your own as well (though there are so many available that there's rarely a case for that, honestly).

Useful Contrib Fields (and their modules)

- Date (date)
- Date Popup (date)
- Media (media)
- Media YouTube (media_youtube)
- Entity Reference (entityreference)

These are just a few of the available field types that come from contributed modules, but they are some of my favorites. Indeed, I'd argue that at least some of these are pretty much essential.

Aw, CRUD!

- **C**reate
- **R**ead
- **U**pdate
- **D**elete

Drupal allows you to store and access information in the database directly – there is lots of functionality devoted to database interactions. So, how are entities different? Why would we use them instead of just storing information in the DB?

One of the most powerful parts of Drupal has to do with what we call CRUD operations: Create, Read, Update, Delete. All of these entity types automatically define their own HTML forms to do those operations. You don't have to write a whole set of forms, form handling code, validations, database operations, and confirmation messages for each of them. It all happens automatically as soon as you turn on their module(s) and configure the types and fields.

[Discuss:
    * Break out to show Demo Site, edit node
]

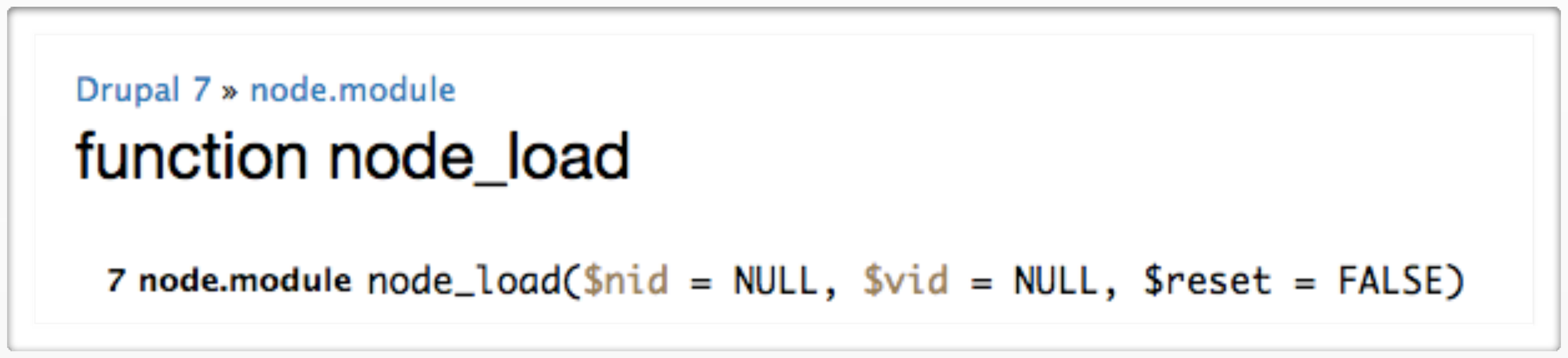
Using Entities and Fields in Code

Saturday, October 24, 15

So, most of this is pretty basic stuff that you're probably already using. So why am I showing it?

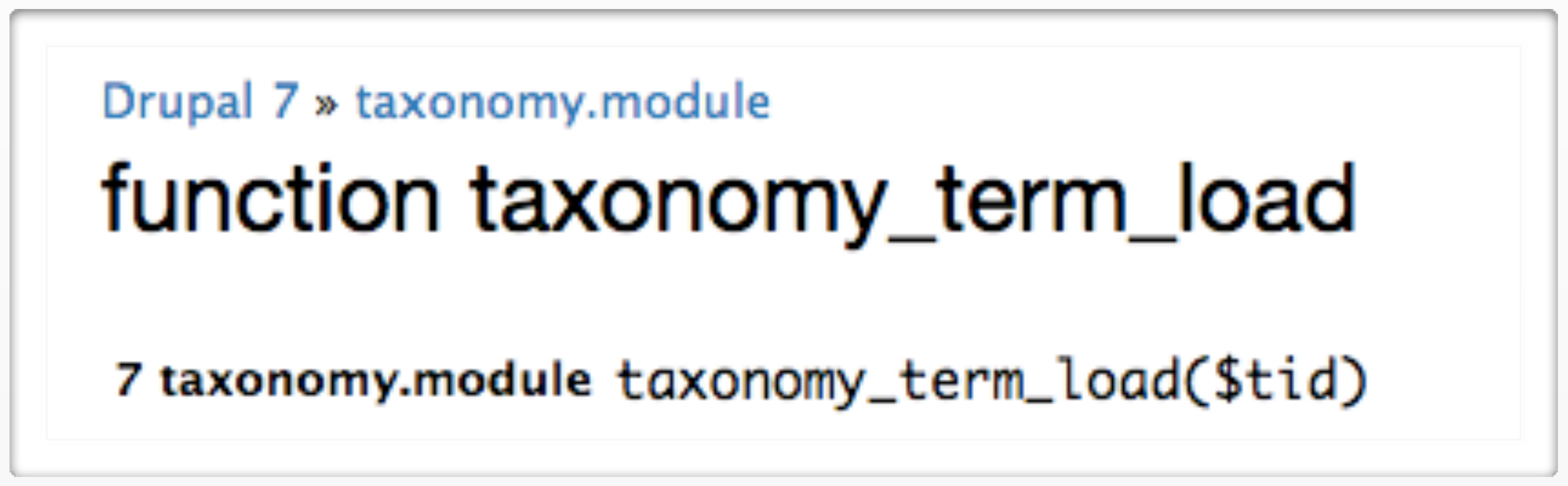
Because all of this stuff exists in the code, too. Entities are standardized in their User Experience, but also on the code side too.

entity_load


Drupal 7 » node.module

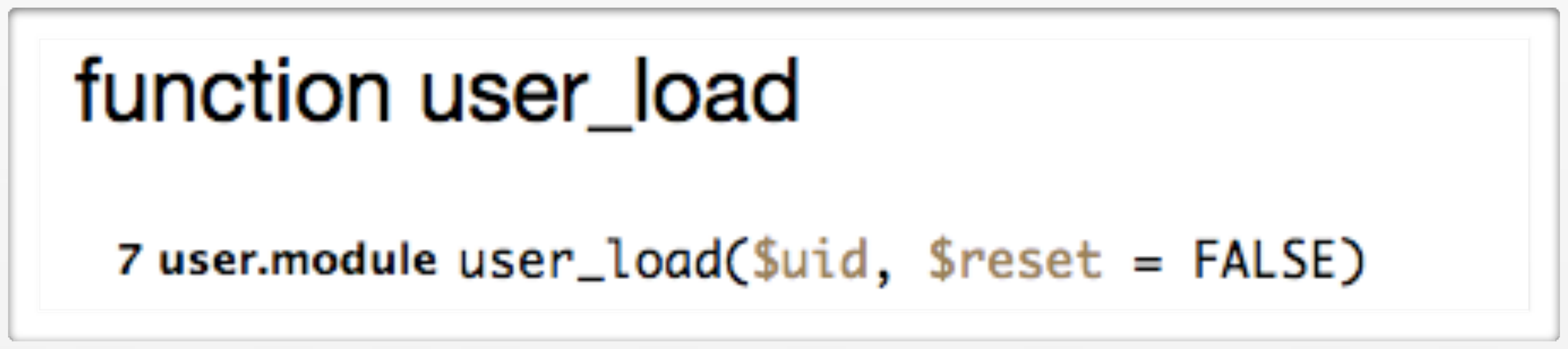
function node_load

```
7 node.module node_load($nid = NULL, $vid = NULL, $reset = FALSE)
```

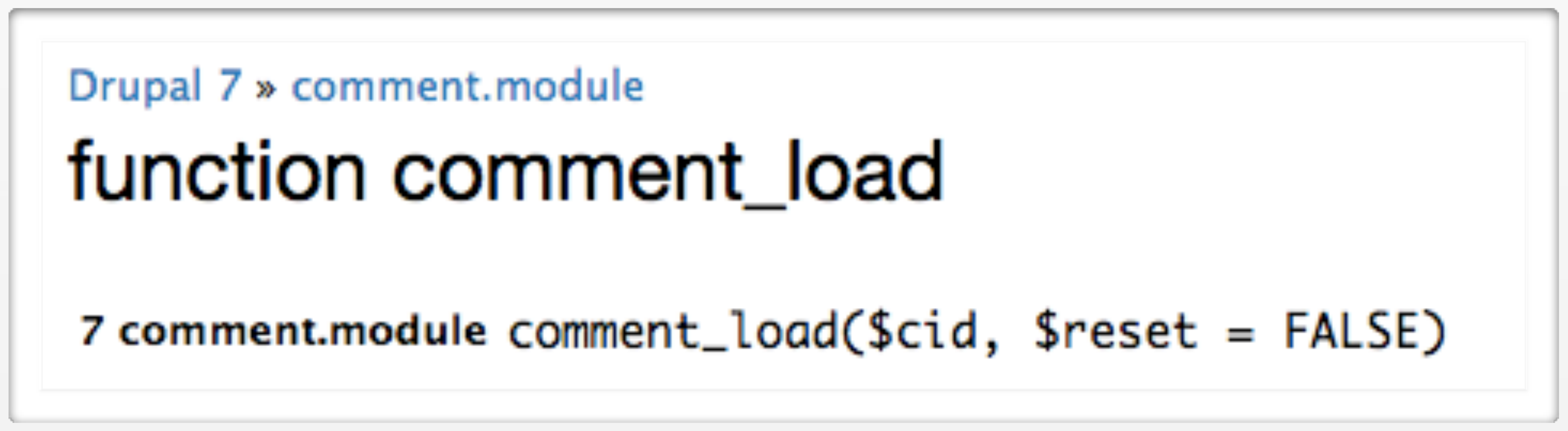

Drupal 7 » taxonomy.module

function taxonomy_term_load

```
7 taxonomy.module taxonomy_term_load($tid)
```

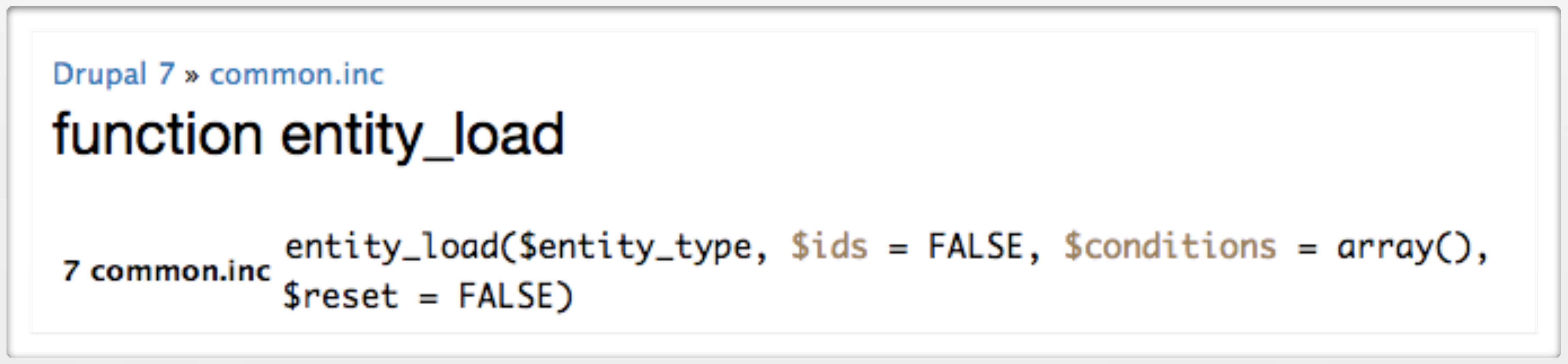

function user_load

```
7 user.module user_load($uid, $reset = FALSE)
```


Drupal 7 » comment.module

function comment_load

```
7 comment.module comment_load($cid, $reset = FALSE)
```


Drupal 7 » common.inc

function entity_load

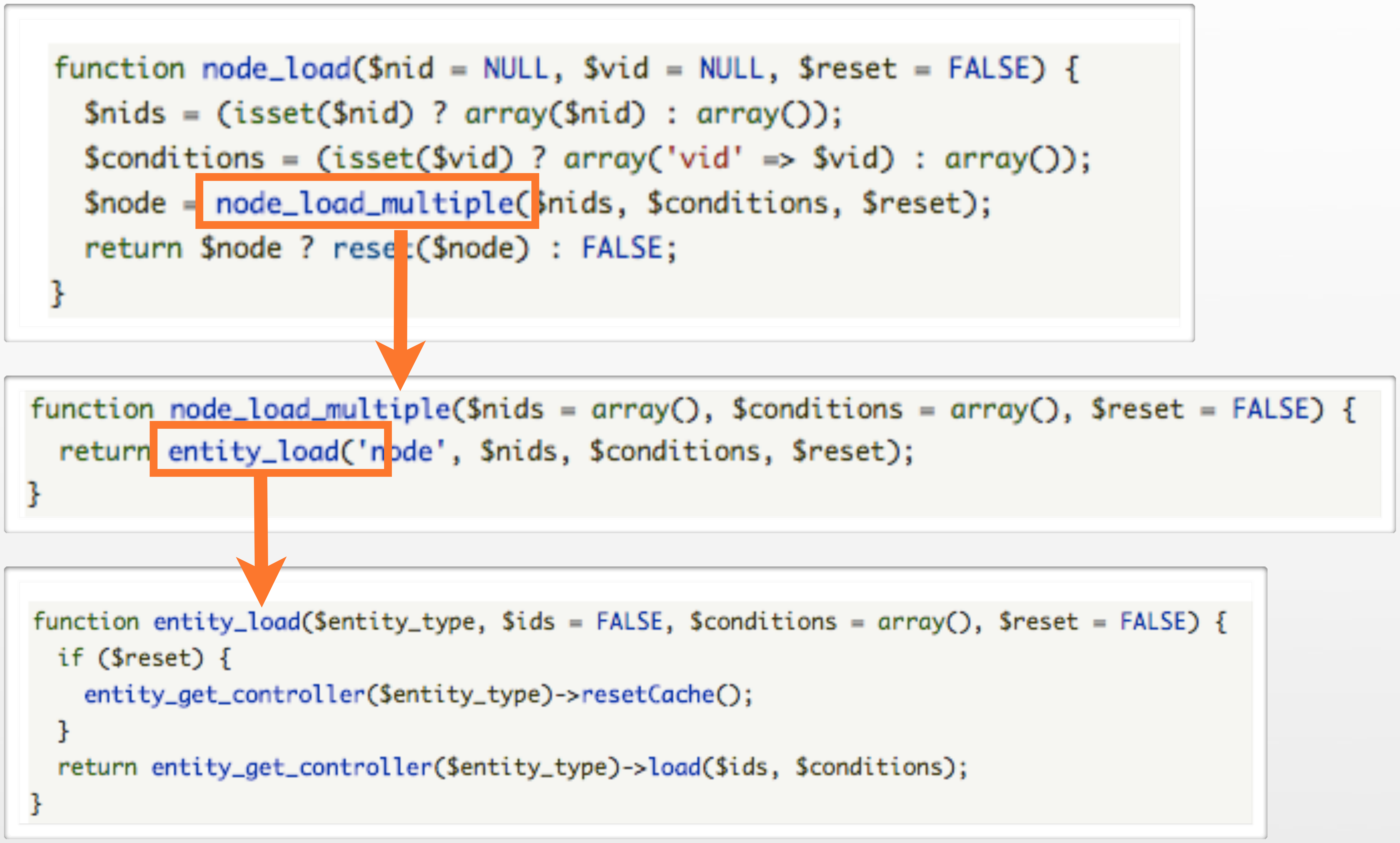
```
7 common.inc entity_load($entity_type, $ids = FALSE, $conditions = array(),
             $reset = FALSE)
```

Want to load an entity in code? Use [entityname]_load, and give it the id you want to load. All entity types have a load function.

Now, what do I mean by “load” an entity?

Loading an entity means, querying the database for one or more blocks of content, by id number. Drupal goes and gets the all the data from all the fields and tables related to that block of content, and assembles them as a PHP object, so that you can do whatever you need to do. This is the “R” – Read – in CRUD.

node_load



```
function node_load($nid = NULL, $vid = NULL, $reset = FALSE) {
  $nids = (isset($nid) ? array($nid) : array());
  $conditions = (isset($vid) ? array('vid' => $vid) : array());
  $node = node_load_multiple($nids, $conditions, $reset);
  return $node ? reset($node) : FALSE;
}
```

```
function node_load_multiple($nids = array(), $conditions = array(), $reset = FALSE) {
  return entity_load('node', $nids, $conditions, $reset);
}
```

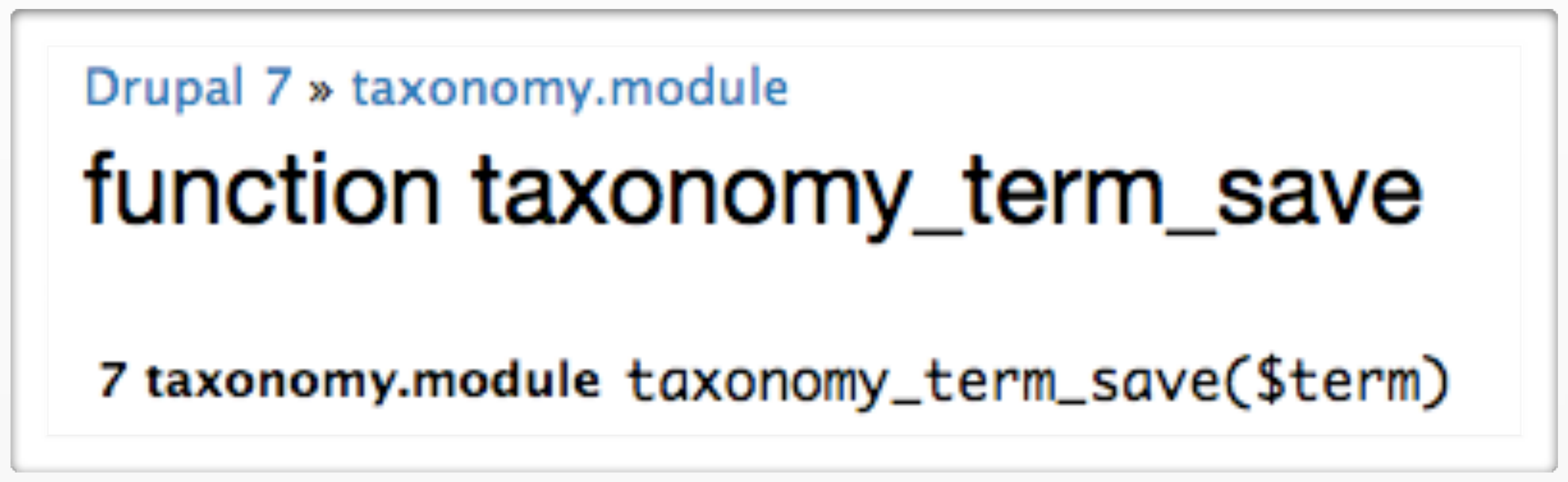
```
function entity_load($entity_type, $ids = FALSE, $conditions = array(), $reset = FALSE) {
  if ($reset) {
    entity_get_controller($entity_type)->resetCache();
  }
  return entity_get_controller($entity_type)->load($ids, $conditions);
}
```

Saturday, October 24, 15

And all entity load functions implement the core function "entity_load".

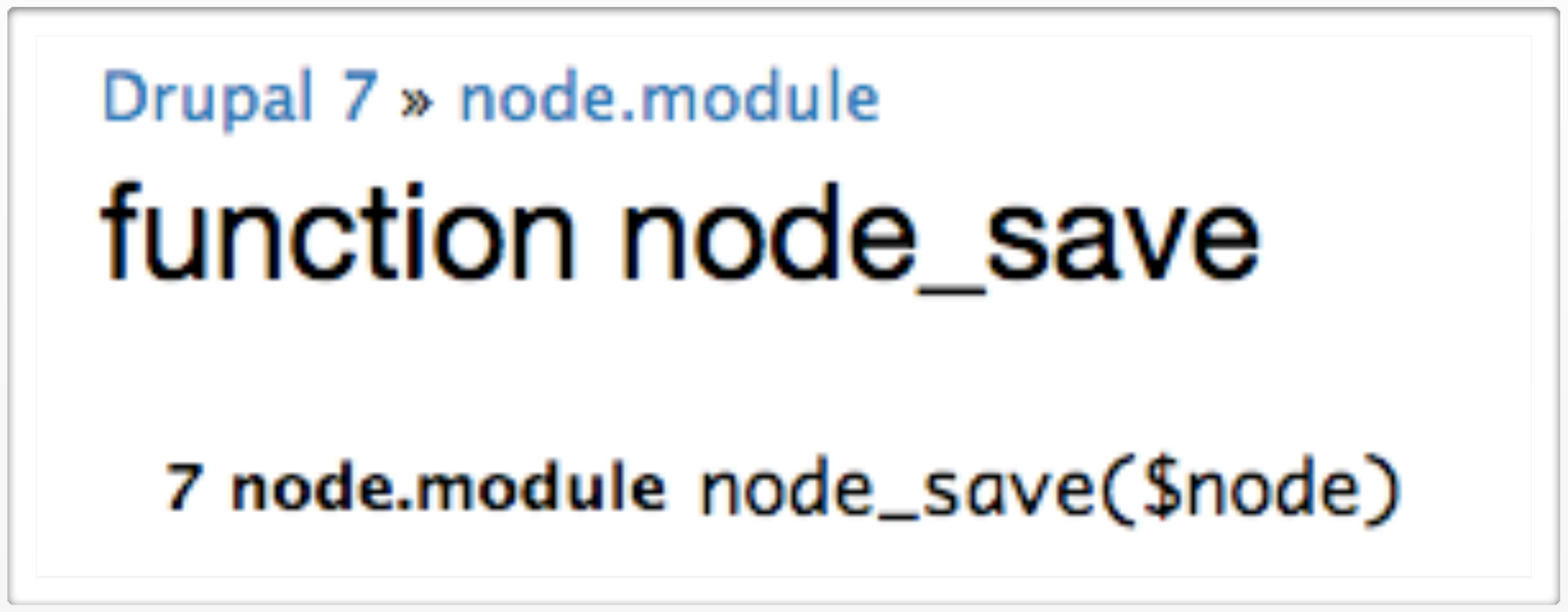
Sometimes there's a layer between them, but ultimately all entities work this way.

[entity]_save


Drupal 7 » taxonomy.module

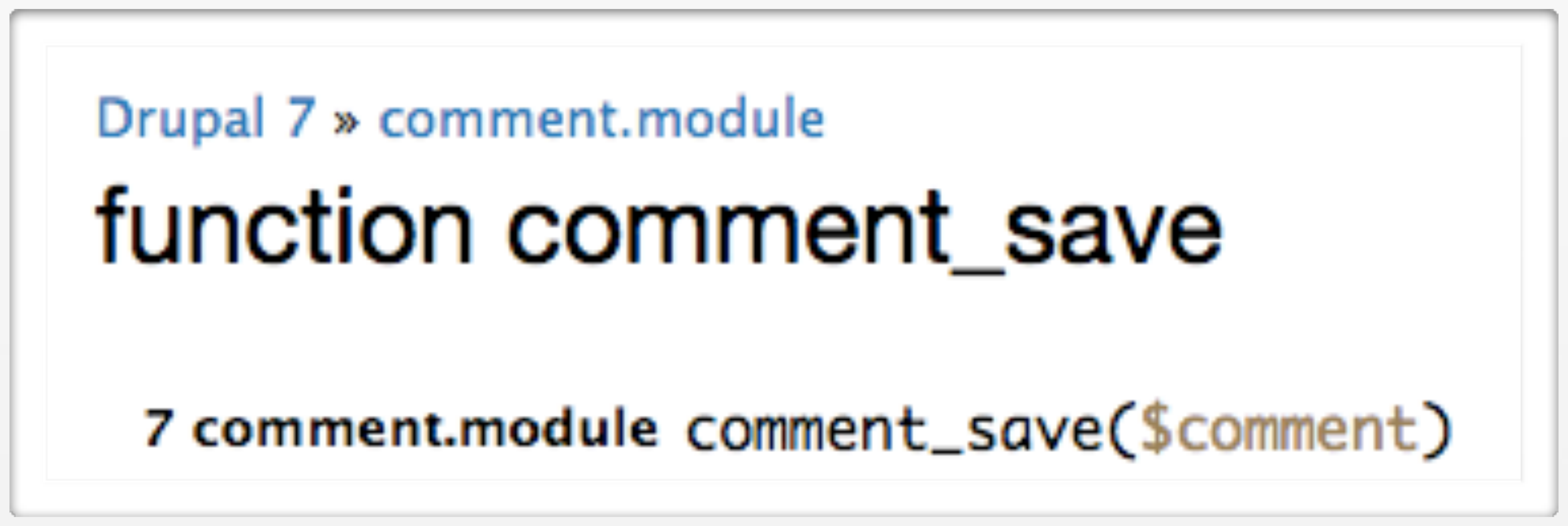
function taxonomy_term_save

7 taxonomy.module `taxonomy_term_save($term)`


Drupal 7 » node.module

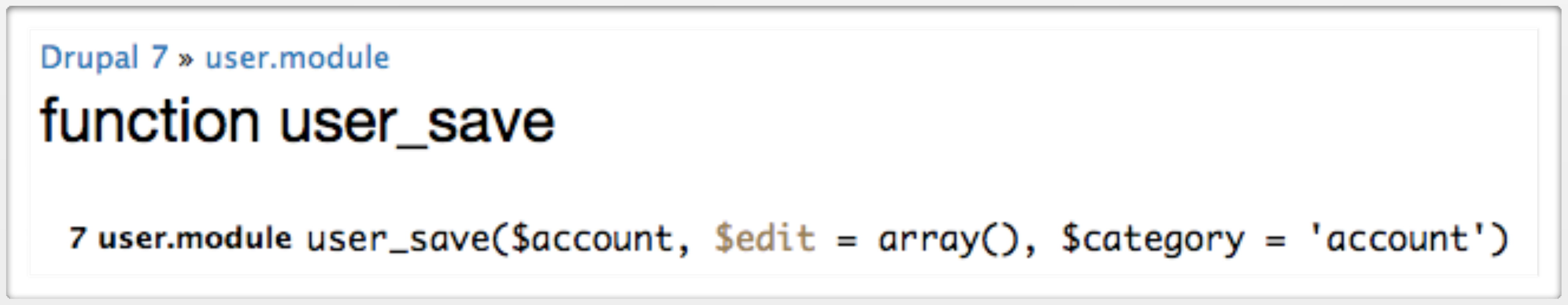
function node_save

7 node.module `node_save($node)`


Drupal 7 » comment.module

function comment_save

7 comment.module `comment_save($comment)`


Drupal 7 » user.module

function user_save

7 user.module `user_save($account, $edit = array(), $category = 'account')`

Saturday, October 24, 15

Saving an entity – either a new one or one that already exists, is much the same.

A new one would be the “C” in CRUD, for “Create”, while an existing one would be the “U”, for “Update”.

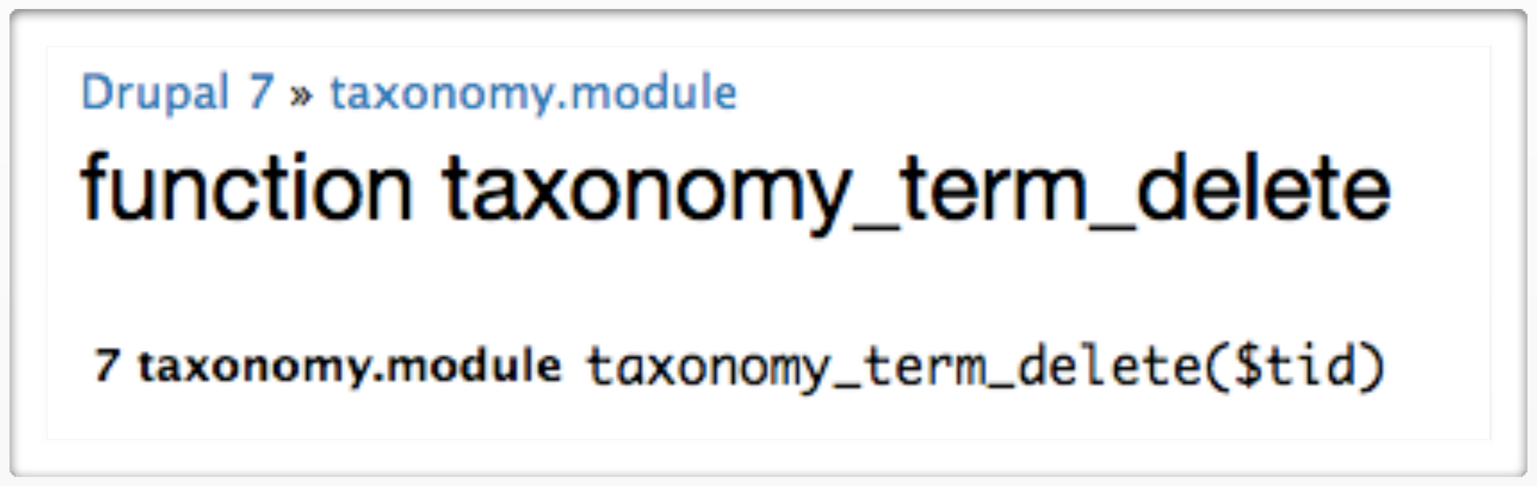
Each entity type has an [entityname]_save function, though it’s worth noting that, at least in Drupal 7, they actually do not call a centralized entity_save function.

However, they all work more or less the same way; you load or create a PHP object with the right values, then call the function and pass it that object. The difference between Creating and Updating is whether or not you have passed it an ID value.

So, what can we do with it?

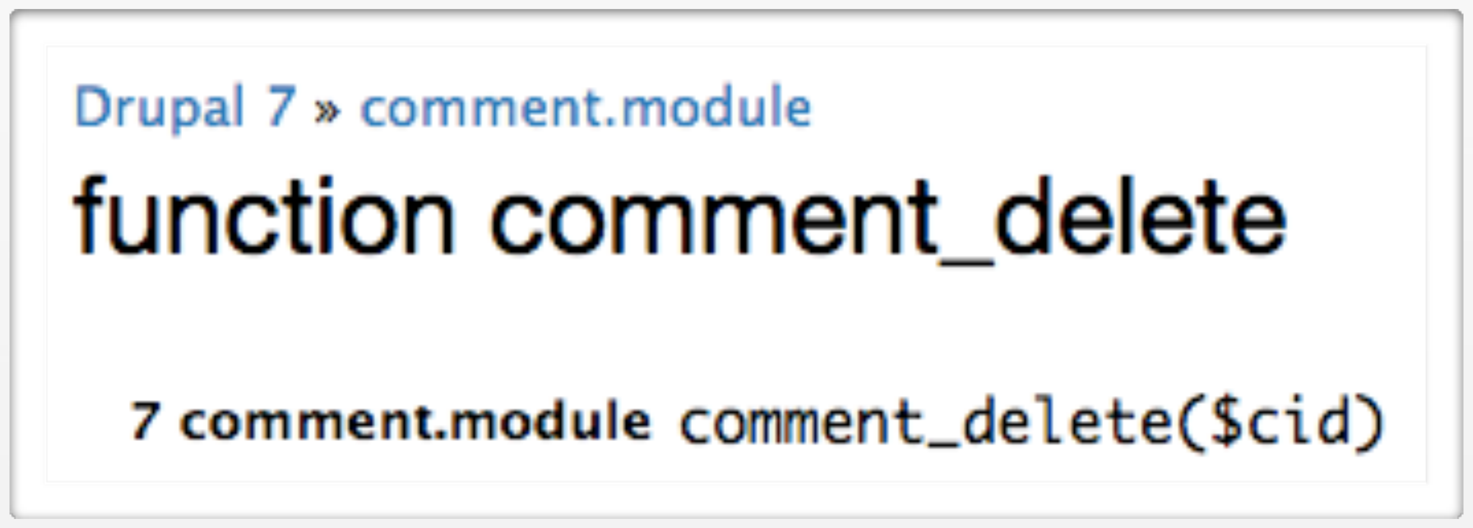
[Discuss:
    * Breakout and examine the module_demo_counter().
    * Add dpm to the function so we can see the entity.
]

[entity]_delete


Drupal 7 » taxonomy.module

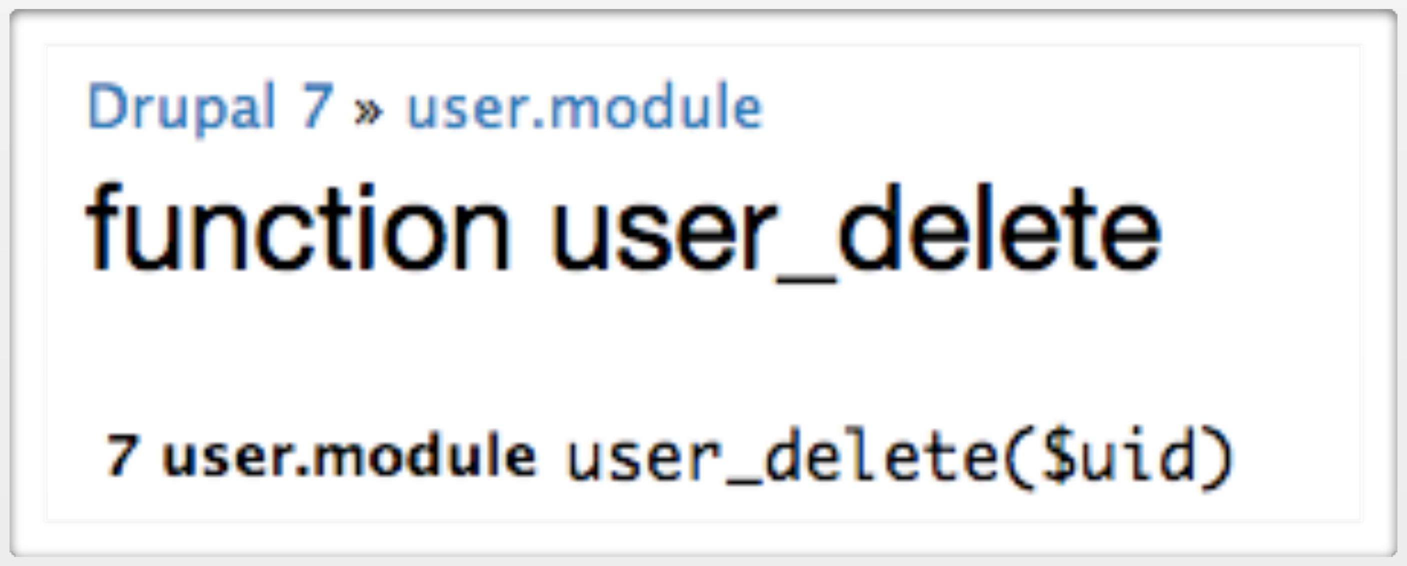
function taxonomy_term_delete

7 taxonomy.module `taxonomy_term_delete($tid)`


Drupal 7 » comment.module

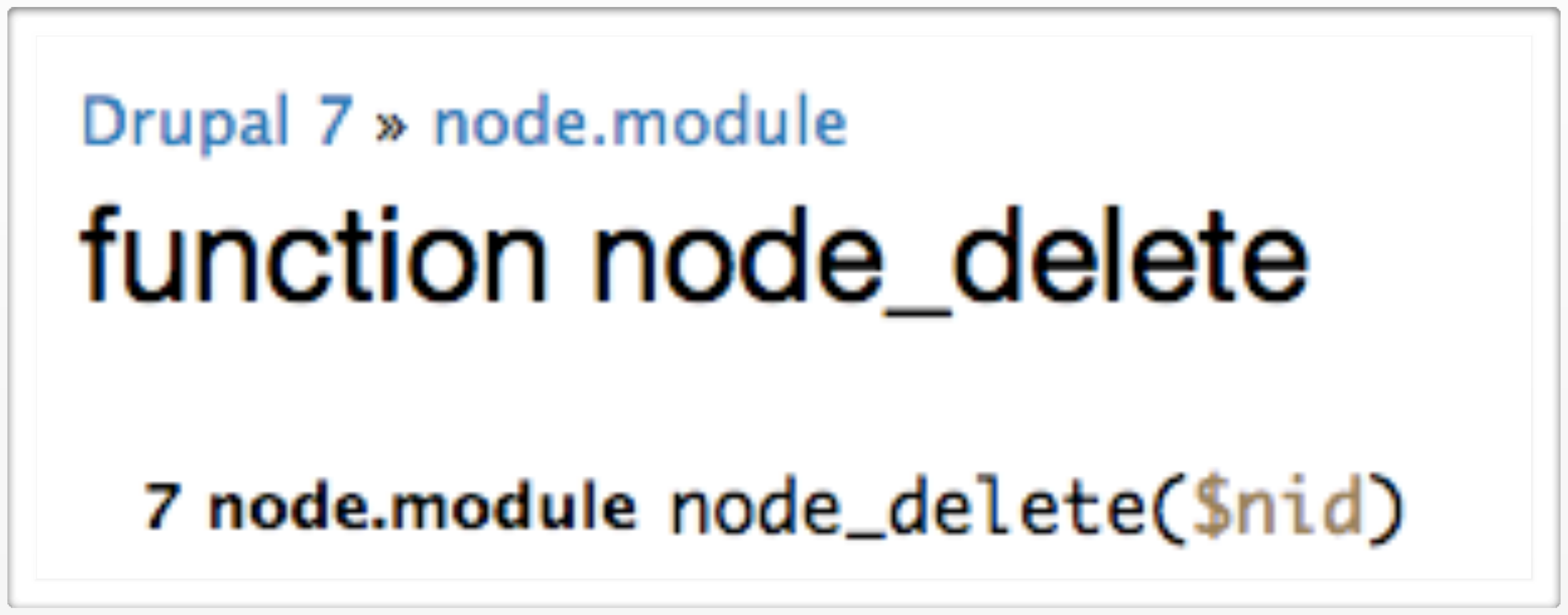
function comment_delete

7 comment.module `comment_delete($cid)`


Drupal 7 » user.module

function user_delete

7 user.module `user_delete($uid)`


Drupal 7 » node.module

function node_delete

7 node.module `node_delete($nid)`

Saturday, October 24, 15

Deleting an entity – the “D” in CRUD – is also very similar.

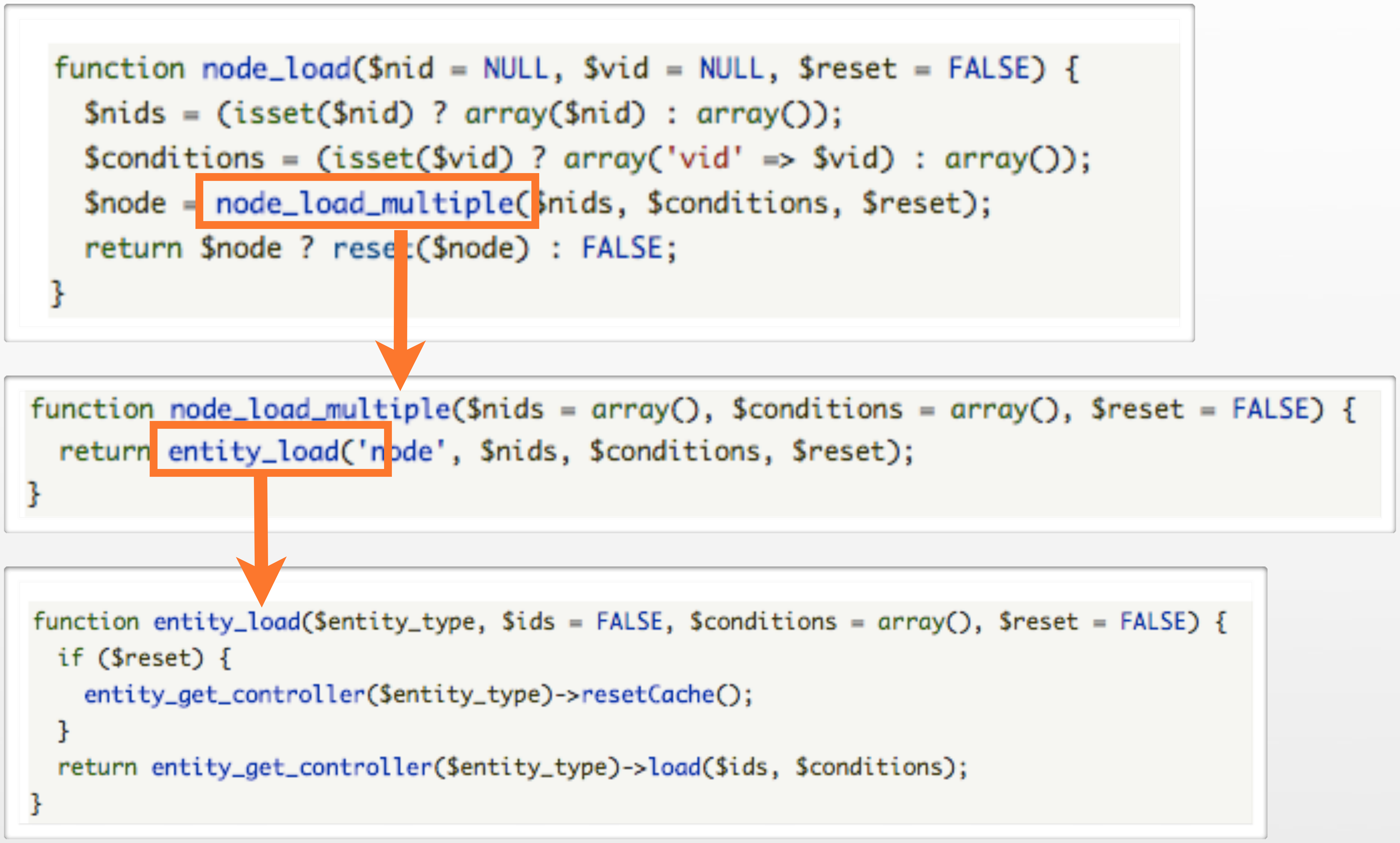
Each entity type has an [entityname]_delete function, though again, at least in Drupal 7, they actually do not call a centralized entity_delete function.

I’m not going to demo that, but it works just like a load function.

One important thing, though – when you delete an entity through the user interface, you usually get a confirmation page asking you if you really want to do that.

You do NOT get that with code. Delete a node with node_delete and it’s gone. Hope you have a backup.

[entity]_load_multiple



```
function node_load($nid = NULL, $vid = NULL, $reset = FALSE) {
  $nids = (isset($nid) ? array($nid) : array());
  $conditions = (isset($vid) ? array('vid' => $vid) : array());
  $node = node_load_multiple($nids, $conditions, $reset);
  return $node ? reset($node) : FALSE;
}
```

```
function node_load_multiple($nids = array(), $conditions = array(), $reset = FALSE) {
  return entity_load('node', $nids, $conditions, $reset);
}
```

```
function entity_load($entity_type, $ids = FALSE, $conditions = array(), $reset = FALSE) {
  if ($reset) {
    entity_get_controller($entity_type)->resetCache();
  }
  return entity_get_controller($entity_type)->load($ids, $conditions);
}
```

Saturday, October 24, 15

I want to draw your attention now to something you may have noticed earlier – the “node_load_multiple” function. Most entities have functions like this one, which allow you to load *more than one* entity at a time.

In many cases, as we see here, they are used as an intermediary step between a single-loader function (node_load) and the entity_load function. It’s a shortcut, essentially. entity_load can take multiple ids (as an array) and return multiple results (also as an array). node_load and its cousins are meant to return only a single result, so they basically make use of their multiple load functions, but only pass them a single id value.

Interestingly, though, there’s nothing that is stopping you from using the [entity]_load_multiple functions yourself. Let’s take a peek at what that looks like.

[Discuss:
        * module_demo_counter_multiple()
        * performance concerns on loading many entities
]

Views & EFQ

Saturday, October 24, 15

Ok - so, we've seen a fair bit of what you can do with Entities and Fields.

Let's talk about lists.

There's basically two ways to do it in Drupal - Views, and Entity Field Query. They are vastly different, both very powerful, and have a lot of overlap in what they can do.

Views

Saturday, October 24, 15

Views is the single most popular contributed modules in Drupal, according to statistics posted on drupal.org.

Views is, at its heart, a query generator - it generates a database query that pulls a list of entities, which can be filtered by any value in the entity.

So, you can use it to generate a list of all nodes of type “blog”, or all users born on your birthday, or all nodes tagged with a specific taxonomy term.
Probably the most appealing part of Views is that there is a UI for it - you can create a list of content without writing a single line of code.

Let’s take a look at a list like that.

[Discuss:
	* Breakout to views admin, make character view with exposed filter on age range.
]

Views is good at...

- Making lists of content with any/all of their field data.
- Printing those lists in multiple places and ways (pages, blocks, feeds).
- Filtering, especially based on user choices (exposed filters).

Views is *not* so good at...

- Being accessible and easily modifiable in code.
- Doing complex, inter-related queries.
- Performance.

Entity Field Query (EFQ)

Entity Field Query, usually known as EFQ, is another way of generating a list, but this one has no UI. It's all in code.

That does make it a little more advanced to use, in some ways, but it is pretty straightforward and there's some great references out there. My personal favorite was written by my friend and colleague, Tim Cosgrove, on the Phase2 blog.

[Discuss:
 * Breakout to module, show module_demo_efq() - character list with filtering on gender.
]

EFQ is good at...

- Making lists of content by id.
- Working with in code.
- Filtering, based on values from code.
- Performance.

EFQ is *not* so good at...

- Being accessible and easily modifiable in a UI.
- Easily displaying output in pages or blocks (possible, but not as easy; requires theming in code).

Joshua Turton

Senior Developer

Email: jturton@phase2technology.com

Twitter: @sjinteractive and @phase2

Drupal.org: srjosh

Phase2

PHASE2TECHNOLOGY.COM