Representations

Data from the Omeka S API (which covers just about all data you'll deal with that's stored in Omeka S) is accessed internally through "representation" objects. These are what you'll receive from calls to the API, and what you will use in modules and themes to access and use the data.

Common Representation Methods

Some pieces of functionality are more or less common to all representations. Usually these have to do with basic operations like getting URLs or links to the represented resources.

URLs

The methods `url`, `adminUrl`, and `siteUrl` each return a URL to the represented resource in the Omeka S web interface. `adminUrl` always returns links to the admin interface, `siteUrl` returns links to the public interface (if the resource has no public URL, it will return `null`), and `url` returns links to the current context, admin or site, as appropriate.

Typically you should use `url`. `adminUrl` and `siteUrl` are generally used only when making URLs to traverse from the admin to public sides or vice-versa, or from one site to another.

`url` and `adminUrl` take the "action" to link to as the first argument. If omitted or set to `null` explicitly, the URL will be to the "default" action, which is usually to simply show or view the resource. `siteUrl` always returns links to the default action, and instead optionally takes the slug of the site to link to as its first argument (the current site will be used if omitted or set to `null`).

All three take the same second argument, a boolean flag for whether the returned URL should be absolute or server-relative. Server-relative URLs are the default and you must pass `true` as this second argument to return an absolute URL instead.

```
$item->url(); // URL to an item's default "show" page

$item->adminUrl('edit'); // URL to an item's "edit" page on the admin, even if
called on the public site

$item->siteUrl('example', true); // Absolute URL to an item's "show" page on
the site "example"
```

There is also an `apiUrl` method for getting a URL to the resource in the REST API. It takes no arguments.

Links

Building upon the methods shown above to get URLs, representations also have some convenience methods for creating HTML links using those URLs.

`link` returns an `<a>` tag with specified text content. The text content is the first argument and is mandatory. You can optionally specify an action to link to (passed through to `url`) as the second argument, and an array of attributes to set on the `<a>` tag as the third argument. The text content will be automatically HTML-escaped (so things like HTML tags within the text will be displayed literally to the user and not interpreted as actual tags).

`linkRaw` works exactly as `link` does and takes the same arguments, but the content is *not* escaped automatically. This allows you to include things like images and other HTML in the link, but you are responsible for filtering or escaping any data coming from a user, including data stored in the database. Using other methods as the input to this function that themselves produce "safe" output is usually a good idea.

```
echo $item->link('View Item'); // Print a link to $item

echo $item->link('Edit Item', 'edit'); // Print a link to the edit form for
$item

echo $item->link('View Item', null, ['title' => 'Not an interesting example,
sorry']); // Print a link to $item with the given title attribute

echo $item->linkRaw($this->thumbnail($item, 'medium')); // Print a thumbnail
for $item that links to the item's page
```

An additional method, `linkPretty`, is only available for Resource-type representations: those for items, item sets, and media. See below for information on that method.

Primary Media

Many different types of resources in Omeka S can have a "primary" media, the one used when Omeka S needs to show a thumbnail for the resource. Items return their first media, media return themselves, and item sets return the primary media for their first item.

The `primaryMedia` method takes no arguments and returns the primary media, itself a Representation. If there is no primary media for the resource, it returns `null`.

Others

There are several more commonly-available methods, like `id` to get the resource's ID, `jsonSerialize` to output the Omeka S API JSON-LD for the resource, and `embeddedJsonLD` for getting a `script` tag for embedding that JSON-LD on an HTML page. Generally these are of internal interest and usage, but depending on what you're doing they might be relevant.

Resource Representations

The most commonly used representations are for the "Resource" types: items, item sets, and media. These are the objects that hold all the values for properties and they accordingly share a great deal of functionality for accessing and printing that data.

Links (again)

The resource representations share the basic link methods that any resource has, described above, but they also have access to one more.

The `linkPretty` method returns a link to the resource consisting of a thumbnail together with the resource's title. This is such a common pattern throughout Omeka S and many themes that we provide a method just for convenience's sake.

All the arguments to `linkPretty` are optional:

- The first is the type of thumbnail to use. `square` is the default.
- The second is the "default" text to use if the resource has no title. If omitted the standard `"[Untitled]"` text will be used.
- The third is the action to link to. This works exactly as it does in the normal linking and URL methods: if omitted the default action is used, usually a show page.
- The final argument is an array of attributes to set. `resource-link` will always be appended to the `class` attribute, even if you set one here.

Displaying Metadata Values

Title and Description

Title and Description are so commonly used in the interface and on themes, there are special methods just for printing them and providing defaults if they are missing.

`displayTitle` will return the first `dcterms:title` value for the resource. If there is no such title, it will display a default value instead. The "default default" is the text `[Untitled]`, translated into the current locale. (Media will fall back to their "source" name instead.) If you want a different default or fallback value, you can provide it as the only argument to this method.

`displayDescription` does the same as `displayTitle`, but with `dcterms:description`. One slight difference: the "default default" for this method is just `null`. So, if you don't pass your own fallback value, printing `displayDescription` will just print nothing.

Any Property

For getting specific metadata values other than title and description, you will use the `value` method.

`value` takes one mandatory argument, the property term to retrieve values for, and one optional argument, an array of options.

The term is a colon-separated string consisting of the namespace prefix of the vocabulary and the local part of the property you want to retrieve values for. For example, `dcterms:subject` or `foaf:name`.

The options array is an associative array with several possible keys for different options:

- `'type'`: pass `'literal'`, `'uri'`, or `'resource'` here to restrict to the given type of values. By default there is no restriction on the types of values that will be included.
- `'all'`: pass `true` here to return *all* the resulting values as an array. By default only the first value will be returned, directly.
- `'default'`: the value passed here will be returned if there are no results
- `'lang'`: pass a language code here to restrict the results to only values matching that langauge. By default there is no restriction on the languages of values.

The values returned by this method are actually ValueRepresentation objects (or an array of those objects, if the `'all'` option was used). To display them on a page, the simplest option is to use the `asHtml()` method, which will take care of escaping text properly, as well as displaying more complex values like links or data types added by modules.

```
echo $item->value('dcterms:subject'); // Print the first Dublin Core Subject
for $item

// Print all the Subject values marked as being in Spanish
$values = $item->value('dcterms:subject', ['all' => true, 'lang' => 'es']);
foreach ($values as $value) { echo $value->asHtml(); }
```

For "resource" type values (links to other Omeka S resources), `asHtml()` will produce a link to the resource. If you're looking to do more than simply make the default link, the method `valueResource()` will return the resource the value is linking to. Then you can use that result like any other item, set, or media representation. `valueResource()` returns null for other kinds of values or if the linked resource can't be found.

```
// Print the creator of a resource linked via the Dublin Core Relation
property
$relatedResource = $item->value('dcterms:relation')->valueResource();
if ($relatedResource) {
    echo $relatedResource->value('dcterms:creator');
}
```

All Properties

Often on "show" pages for resources, you want to display *all* the values, no matter their property. The method `displayValues` provides this functionality.

`displayValues` optionally takes an array of options as its only argument. There is currently one option:

- `'viewName'` : the name of a different partial to use than the default

The output returned by `displayValues` is themable: the partial is `common/resource-values.phtml`.

Other Resource Data

Beyond the values, there is other common resource data you might need to access, and each piece generally has its own method.

- `resourceClass` and `resourceTemplate` get you the Representations for the class and template for this resource, if any, respectively. To more simply print the resource class's label, there is a convenience method `displayResourceClassLabel` . It takes an optional default fallback value, like `displayTitle` and `displayDescription` .
- `owner` gets the Representation for the user that owns the resource, if any.
- `isPublic` returns a boolean marking whether the resource is public.
- `created` and `modified` return PHP DateTime objects for the dates the resource was created and last modified, respectively. The `$this->i18n()->dateFormat()` helper is useful for printing and localizing dates.
- `values` simply returns an array of all the values for the resource, grouped by property, with the properties' Representations and any alternate labels or comments imposed by the resource template included.
- `subjectValues` , `subjectValueTotalCount` , and `subjectProperties` are used by the interface for showing values that link to this resource.

Item-specific methods

- `media` returns all the media assigned to the item, as an array of MediaRepresentations
- `itemSets` returns all the item sets the item is in, as an array of ItemSetRepresentations

Item set-specific methods

- `itemCount` returns the number of items in the set
- `isOpen` returns a boolean indicating if the set is open for additions

Media-specific methods

- `render` returns HTML for displaying the media. An array of options can be passed.
- `originalUrl` returns the URL to the original version of the file in Omeka S's file storage (if there is an original file)
- `thumbnailUrl` returns the URL to a thumbnail of the media. The type of thumbnail must be passed as the only argument. A fallback thumbnail URL will be returned if the media has no thumbnails stored. (Users should generally use the `thumbnail` view helper rather than this method directly.)
- `thumbnailUrls` returns an array of all the thumbnail URLs for this media, with the keys being the thumbnail types and the values being the URLs.
- `ingester` returns the ingester used when adding this media. `ingesterLabel` returns a human-readable label for the ingester.
- `renderer` returns the renderer used for this media
- `mediaData` returns additional data stored for the media. This differs from media type to media type, and many store no extra data at all.
- `source` returns the source of the media, usually a filename or URL it was retrieved from
- `mediaType` returns the media type (also known as MIME type) for the original file, if any
- `sha256` returns the SHA-256 hash of the original file, if any
- `size` returns the size of the original file, if any, in bytes (added in Omeka S 1.2.0)
- `storageId` and `extension` return the randomized filename and file extension used for the file in Omeka storage, if any. `filename` returns these two combined with a period, forming the full as-stored filename. `storageId` may exist for media with no original file stored if there are thumbnails
- `hasOriginal` returns a boolean indicating whether an original file is stored for this media
- `hasThumbnails` returns a boolean indicating whether thumbnails are stored for this media
- `lang` returns the declared language code for this media, if any
- `item` returns the parent item of this media, as an ItemRepresentation