PHP API

From within Omeka's PHP environment there are three ways to make API requests: directly, using the API manager service, and indirectly, using the API controller plugin or the API view helper.

API Manager

You can perform API operations directly using Omeka's API manager. The API manager is responsible for authorizing the user, initializing the API request, performing the operation on the resource, and finalizing the API response. (Note that not all resources implement every API operation.)

You can access the API manager directly using the service locator (should you have access to the service locator, of course). Once you have the manager you can perform the following API operations:

- `search($resource, $data, $options)` : search resources by query
- `read($resource, $id, $data, $options)` : read a resource by ID
- `create($resource, $data, $fileData, $options)` : create a resource
- `batchCreate($resource, $data, $fileData, $options)` : batch create resources
- `update($resource, $id, $data, $fileData, $options)` : update a resource by ID
- `batchUpdate($resource, $ids, $data, $fileData, $options)` : batch update resources by IDs
- `delete($resource, $id, $data, $options)` : delete a resource by ID
- `batchDelete($resource, $ids, $data, $options)` : batch delete resources by IDs

The `$resource` argument is the resource name. For read, update, and delete operations, the `$id` argument is the resource ID. The `$data` argument is an optional array of API request parameters. For create and update operations, the `$fileData` argument is an optional array of file data. The `$options` argument is an optional array of API request options.

Each operation will return an API response object containing the requested data and other relevant information. You can get the resultant representation by calling `getContent()` on the response. For example, to get an item with an ID of 123:

```
// Where $services is Omeka's service locator object.
$api = $services->get('Omeka\ApiManager');
```

```
$response = $api->read('items', 123);
$itemRepresentation = $response->getContent();
```

Controller plugin

You can access the API in your controllers using the API controller plugin, accessible within a controller by calling `$this->api()`. The plugin has every operation that the API manager has, plus an additional one for convenience:

- `searchOne($resource, $data, $options)`: search for one resource by query

For example, to get an item representation via a route parameter in a typical show action:

```
public function showAction()
{
    $itemRepresentation = $this->api()->read('items', $this->params('id'))-
>getContent();
    $view = new ViewModel;
    $view->setVariable('item', $item);
    return $view;
}
```

View helper

You can access the API manager in your views using the API view helper, accessible within a view by calling `$this->api()`. The helper only has the `search()`, `searchOne()`, and `read()` operations. This is prevent operations that change state from being executed in the view layer.

For example, to get an item in a view template:

```
<?php $itemRepresentation = $this->api()->read('items', 123); ?>
```

Responses

All API operations return a response object of class `Omeka\Api\Response`. The main method of use is `getContent()`, which returns the actual content that was requested. For `read`, this is a single value, while `search` will have content that is an array.

By default, the values returned by `getContent()` will be representation objects, suitable for use in views and controllers.

Another method is useful for `search` specifically: `getTotalResults()` returns the total number of resources that matched the given query, even those beyond limits or pagination specified in the

request. This "total" value is often presented to the user or used for things like pagination to calculate how many pages of results exist.

Errors

The API methods throw exceptions for error cases. These exceptions all implement the `Omeka\Api\Exception\ExceptionInterface` interface. Common exceptions include:

- `Omeka\Api\Exception\NotFoundException`: when an request for an operation like `read` or `update` specifies an ID that can't be found
- `Omeka\Api\Exception\PermissionDeniedException`: when the current user lacks the necessary level of permissions to fulfill the request
- `Omeka\Api\Exception\ValidationException`: when an `update` or `create` request specifies invalid data. The specific validation error messages are avaiable from the `getErrorStore()` method of the exception.

For the controller helper specifically, a Form object can be passed: `$this->api($form)`, and if a validation error occurs, the helper will try to automatically match up the returned error messages with their corresponding form elements, so the user can see the errors in context, as well as showing them as "flash" messages on the top of the screen.