23. CQRS API guidelines

Date: 2023-08-24

Status

Accepted

Context

In PrestaShop 9.0, a new API is being built powered by ApiPlatform, based on Commands (from CQRS). We need to define a convention for the path format and the ApiPlatform resources classes. We will mostly follow REST conventions but it's useful to write down clearly a few things to avoid confusion later.

Convention

URI path and parameters conventions

We base the naming on the domain from CQRS (which usually matches the ObjectModel entity name as well) in `PrestaShop/PrestaShop/Core/Domain`

Example: Hook

Domain: `PrestaShop/PrestaShop/Core/Domain/Hook`
ApiPlatform resource class: `PrestaShopBundle\ApiPlatform\Resources\Hook`

URI conventions, we use the domain word as a base for the URI, for single entity APIs we use the singular form, for list (or when we return a collection) we use the plural. For the identifier related to one entity we use the domain name suffixed by `Id` (ex: Hook -> `hookId`).

List: `ApiPlatform\Metadata\GetCollection /hooks`
One hook endpoint: `ApiPlatform\Metadata\Get /hook/{hookId}`

For APIs that are sub part of a larger entity (whether it's to display its related entities or small parts of a big entity) we keep the initial domain name as the beginning of the URI and append it with the definition of the sub parts (separated by a `/`). If it's the sub part of an identified entity, we complete the initial URI path and append the sub part after the entity ID. When the sub part is an action name or a compound name we use kebab case convention.

Example:

- Hook status (sub part): `/hook/{hookId}/status`
- Search (action) products: `/products/search`
- Product combinations (sub part): `/product/{productId}/combinations`
- Assign product to category (action): `/product/{productId}/assign-to-category`
- Set product carriers (set a sub part): `/product/{productId}/carriers`
- Attribute group (list): `/attribute-groups`
- Attribute group associated values (list of sub part): `/attribute-group/{attributeGroupId}/attributes`

Multilang

Some entities in PrestaShop have multilang values (like product names, category descriptions, ...), this data must be presented in the API endpoints:

- single entities endpoints return ALL the languages in an associative array indexed by language ID (ex: `{"localizedNames": [1: "english name", 2: "nom français", ]}`)
- list of entities return only one language so multilang fields are returned as strings (ex: `{"name": "english name"}`), the language used by default is the default language configured on the shop but you can specify a `langId` query parameter to fetch another language values

To allow knowing the association between languages and language IDs a `/languages` endpoint will be accessible without any needed permission.

HTTP methods

Read operations use **GET** method
Creation operations use **POST** method (without ID specified)
Update (partial and/or full) operations use **PUT** method (ex: PUT `/product/{productId}`)
Delete operations use **DELETE** method (ex: DELETE `/product/{productId}`)

Bulk operations

Similar convention for list of IDs we use the domain and append `Ids` at the end.

- Bulk delete products: Method DELETE `/products` with `productIds` parameter in request body (array of product IDs)
- Bulk duplicate products: Method POST `/products/duplicate` with `productIds` parameter in request body (array of product IDs)
- Bulk update status: Method PUT `/products/update-status` with `productIds` parameter in request body (array of product IDs)

CQRS Mapping

Providers/processors are implemented as helpers to automatically transform submitted data through the API into CQRS queries/commands, but it needs to match a convention, the parameters in the API request must match the fields from the CQRS command (constructor parameters and/or setters method). The only exception is for the entity ID which must always have the naming `entityId` if the related command/query parameter doesn't match we'll handle a custom mapping but the URI format should respect the convention.

Example:
Command:
`PrestaShop\PrestaShop\Core\Domain\Customer\Group\Command\AddCustomerGroupCommand`
Command fields:

- `localizedNames: string[]`
- `reductionPercent: DecimalNumber`
- `displayPriceTaxExcluded: bool`
- `showPrice: bool`
- `$shopIds: ShopId[]`

POST parameters:

- `localizedNames: string[]`
- `reductionPercent: float`
- `displayPriceTaxExcluded: bool`
- `showPrice: bool`
- `shopIds: int[]`

Scope names

This convention is only for the Core API, if modules wish to define their own API they are free to use any convention for the scopes. We use the entity domain name in its singular form, and append the action, the whole string is written in snake case. Each scope is supposed to represent a single authorized action.

Basic Example:

- Orders read operations: `order_read`
- Orders write operations: `order_write`

In the future we may define some more detailed sub scope but they should follow a similar convention:

Example:

- Orders update address: `order_update_address`
- Orders create invoice: `order_create_invoice`