How to design batch API (General recommendations)

Introduction

These are common recommendations how to design batch API in FOLIO. Each module should decide how to create batch API according to its business model and hardware possibilities. In this document I collect common recommendations and points of care.

Batch API reason

The main reason of batch API is saving http traffic between FOLIO modules, because each http request goes through the Okapi module.

Heavy Partial-Success Batch API

It means that you send a batch of entities to process them and get a response with details according to each entity. (Here is the branch with reference implementation)

URL pattern

"POST app-name/batch/resource-name/operation"

According to idempotent principle of RESTful services create and update batch of entities should use POST method because each call with the same batch of entities can lead to a different result

For example:

POST /source-storage/batch/records/create - create a batch of records

POST /source-storage/batch/records/update - update a batch of records

Statuses

207 Multi-status. Everything is Ok, look into body for details of each entity

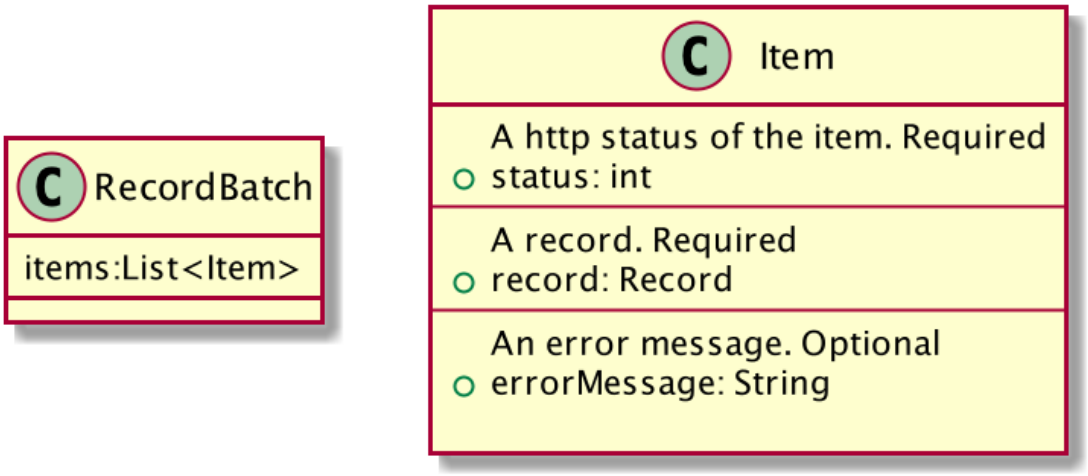
400 Bad Request. Something wrong with URL or params

404 Not Found. Server could not find something according to given data

422 Unprocessable Entity. Body can not be parsed

500 Internal Server error. Something went wrong)

Body schema



Use cases

Fail

- status field describes what kind of error 40X \ 50X.
- record with processed entity
- errorMessage describes what exactly is wrong with current entity.

Success

- status is 20X
- record with processed entity
- errorMessage is empty

Tiny Partial-Success Batch API

It means that you send a batch of entities to process them and get a response with details according to each entity. The main difference with Heavy Partial-Success is that Item doesn't have an entity instead of it here is a href field with relative reference to an entity.

URL pattern

"POST app-name/batch/resource-name/operation"

According to idempotent principle of RESTful services create and update batch of entities should use POST method because each call with the same batch of entities can lead to a different result

For example:

POST /source-storage/batch/records/create - create a batch of records

POST /source-storage/batch/records/update - update a batch of records

Statuses

207 Multi-status. Everything is Ok, look into body for details of each entity

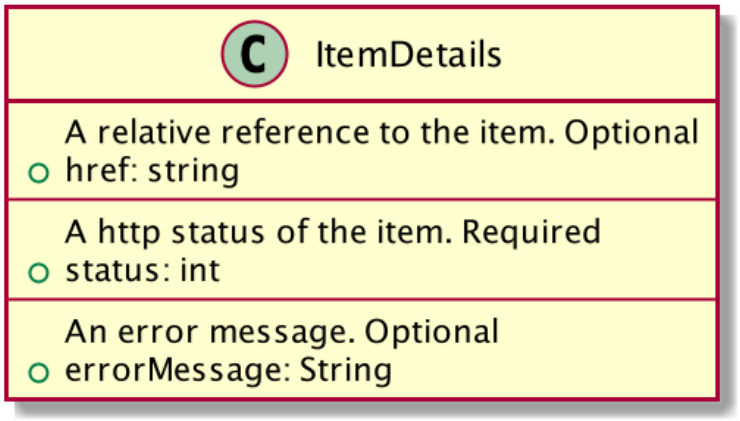
400 Bad Request. Something wrong with URL or params

404 Not Found. Server could not find something according to given data

422 Unprocessable Entity. Body can not be parsed

500 Internal Server error. Something went wrong)

Body schema



Use cases

Fail

- status field describes what kind of error 40X \ 50X.
- href is empty
- errorMessage describes what exactly is wrong with current entity.

Success

- status is 20X
- href contains a relative path to an entity (for example “/source-storage/records/123”)
- errorMessage is empty

Heavy Transactional Batch API

It means either all or nothing.

URL pattern

"POST app-name/batch/resource-name/operation"

According to idempotent principle of RESTful services create and update batch of entities should use POST method because each call with the same batch of entities can lead to a different result

For example:

POST /source-storage/batch/records/create - create a batch of records

POST /source-storage/batch/records/update - update a batch of records

Statuses

200 Ok. Everything is Ok

400 Bad Request. Something wrong with URL or params

404 Not Found. Server could not find something according to given data

422 Unprocessable Entity. Body can not be parsed

500 Internal Server error. Something went wrong)

Body schema

Use cases

Behavior as plain REST endpoint. A list of entities is sent to the server, in success story it gets a list of ids of entities those were processed, otherwise 40X or 50X with error message.

Tiny Transactional Batch API

It means either all or nothing.

URL pattern

"POST app-name/batch/resource-name/operation"

According to idempotent principle of RESTful services create and update batch of entities should use POST method because each call with the same batch of entities can lead to a different result

For example:

POST /source-storage/batch/records/create - create a batch of records

POST /source-storage/batch/records/update - update a batch of records

Statuses

204 No Content. Everything is Ok

400 Bad Request. Something wrong with URL or params

404 Not Found. Server could not find something according to given data

422 Unprocessable Entity. Body can not be parsed

500 Internal Server error. Something went wrong)

Body schema

Just send a list of entities and get either success response as 204 (no content) or errors

Points of care

- **Size expectations or restrictions.** It should decide specific implementation/module. How many items a module can process and how much time it takes the module to do it. For standard of FOLIO we can not demand it from everybody.
- **Synchronous or asynchronous response.** It depends on business model. As the main reason of batch API in FOLIO is save http traffic around Okapi.
- **Complete or partial success / failure.** It depends on implementation. In case of transactional approach it should be complete success/failure otherwise it should be partial success/failure
- **Database transactions (specific to storage modules)** It depends on requirements. For data-import it's necessary to determine which record is not saved during import large files and in this case it should be persisted as error (for further analysing)
- **Streamed processing of records** TBD It's applicable for a document but what would we do in case of json entities? How to unmarshal bytes into entities
- **Number of parallel database connections.** If the maximum number of connections the database accepts is reached all modules and all tenants are blocked that need a new connection. See MODINVSTOR-330 - Getting issue details... STATUS