Content brokering API

Version **DRAFT**, last modified: 2016-01-21 by Teemu Sirkiä

General description

Content brokering API is used to retrieve the available contents of a content server. API is based on HTTP GET requests which return JSON objects.

In this documentation, we assume that the content server main API endpoint is:
http://our.contentserver.example/api/v1

All the requests begin with this common root for all API endpoints.

The cardinality in the tables below have the following meaning:

- 1 = there is always exactly one element or the attribute is always present
- ? = the element may exist or the attribute may be present
- * = there can be zero or many elements

Hierarchical structure of a content server

A content server hosts content
- A content server may have multiple **content types**
  - Content types consist of **content packages**
    - Content packages contain **content items**
      - (Content item may have **subitems**, subsubitems etc.)

Custom attributes

This documentation describes the minimum level of attributes that should exist. Depending on the implementation and the content types available, API endpoints can return additional custom attributes. These attributes are defined in the specific API description of that server.

Fetching available content types

This API endpoint lists all the available content types. In case that the content server hosts only one type of content, there will be only one content type in the response.

http://our.contentserver.example/api/v1/content

An example:

```
{
   "contentTypes":[
      {
         "name":"videos",
         "url":"http://our.contentserver.example/api/v1/content/videos",
         "title":"Video Collection",
         "description":"This is a collection of educational videos.",
      }
   ]
}
```

Root element

`contentTypes`, a list containing `contentType` elements

Attributes of `contentType`

Name	**Type**	**Cardinality**	**Description**
name	string	1	A unique name of this content type
url	string	1	An API endpoint for getting more information
title	string	1	A short name describing the content type
description	string	1	A longer description of the content type

Fetching content packages

This API endpoint lists all the available content packages for a specific content type.

http://our.contentserver.example/api/v1/content/contentType where `contentType` is one of the available content types in the content server.

An example:

```
{
   "contentPackages":[
      {
         "name":"videos-python",
         "url":"http://our.contentserver.example/api/v1/content/videos",
         "title":"Videos teaching programming in Python"
         "description":"This package contains videos teaching programming
           in Python. Videos are designed for novices without any previous
           experience.",
         "author":"Our University",
         "license":"MIT",
         "version":"0.0.1",
         "created":"2015-08-05 13:26:48",
         "modified":"2015-08-05 13:26:48",
         "keywords":["video", "python", "cs1"]
      }
   ]
}
```

Root element

`contentPackages`, a list containing `contentPackage` elements

Attributes of `contentPackage`

Name	**Type**	**Cardinality**	**Description**
name	string	1	A unique name of this content package
url	string	1	An API endpoint for getting more information
title	string	1	A short name describing the content package
description	string	1	A longer description of the content package
author	string	1	The author of this content package
license	string	1	The license to be used with this content type
version	string	1	Version number for this content package
created	string	1	The creation date
modified	string	1	The modification date
keywords	list	1	A list of strings describing the contents

Fetching content items inside the content package

This API endpoint lists all the available content items inside the given content package.

http://our.contentserver.example/api/v1/content/contentType/contentPackage where `contentType` is one of the available content types and `contentPackage` is one of the available content packages in the content server.

An example:

```
{
   "content":[
      {
         "name":"part1",
         "url":
           "http://our.contentserver.example/api/v1/content/videos/part1",
         "html_url":"http://our.contentserver.example/videos/part1",
         "title":"Programming in Python, part 1",
         "description":"This video introduces the basic concepts of
            writing programs in Python",
         "author":"John Doe",
         "version":"0.0.1",
         "created":"2015-08-05 13:26:48",
         "modified":"2015-08-05 13:26:48",
         "keywords":["basics", "variables", "assignment"]
      }
   ]
}
```

Root element

`content`, a list containing `content` elements

Attributes of `content`

Name	Type	Cardinality	Description
name	string	1	A unique name of this content inside the package
url	string	1	An API endpoint for getting more information
html_url	string	1	A URL to see the actual content item
title	string	1	A short name describing the content
description	string	1	A longer description of the content
author	string	1	The author of this content
version	string	1	Version number for this content
created	string	1	The creation date
modified	string	1	The modification date
keywords	list	1	A list of strings describing the content

Fetching more information about one content item

Optional API endpoint

This API endpoint returns more information about the requested content item. The response contains always at least the same information as the previous API endpoint but this endpoint may show more custom data that was not available at the previous API endpoint.

This API endpoint may be left unimplemented if there is no need for it. This API endpoint is useful mainly if the content item has a hierarchical structure that needs to be described.

http://our.contentserver.example/api/v1/content/contentType/contentPackage/content where `contentType` is one of the available content types and `contentPackage` is one of the available content packages and `content` is a content item inside the content package.

An example:

```
{
   "item": {
      "name":"part1",
      "url":
        "http://our.contentserver.example/api/v1/content/videos/part1",
      "html_url":"http://our.contentserver.example/videos/part1",
      "title":"Programming in Python, part 1",
      "description":"This video introduces the basic concepts of
         writing programs in Python",
      "author":"John Doe",
      "version":"0.0.1",
      "created":"2015-08-05 13:26:48",
      "modified":"2015-08-05 13:26:48",
      "keywords":["basics", "variables", "assignment"]
   }
}
```

Root element

`item`

Attributes of `item`

Name	Type	Cardinality	Description
name	string	1	A unique name of this content inside the package
url	string	1	An API endpoint for getting more information
html_url	string	1	A URL to see the actual content item
title	string	1	A short name describing the content
description	string	1	A longer description of the content
author	string	1	The author of this content
version	string	1	Version number for this content
created	string	1	The creation date
modified	string	1	The modification date
keywords	list	1	A list of strings describing the content
subitems	list	?	A list of subitems if available (see the next page), only the first level of subitems is returned

Defining subitems

If the content item has a hierarchical structure, it is possible to describe the structure by using subitems. Subitems can contain more subitems and the depth of the recursive structure is not limited.

Subitems can be used to describe, for example, the chapters or sections inside a video or a book.

An example:

```
{
   "subitem":[
      {
         "name":"chapter1",
         "type":"chapter",
         "url":
           "http://our.contentserver.example/api/v1/
            content/videos/part1/chapter1",
         "keywords":["basics", "variables", "assignment"]
      }
   ]
}
```

Root element

`subitem`

Attributes of `subitem`

Name	Type	Cardinality	Description
name	string	1	A unique name of this subitem inside the content
type	string	1	A string describing the type of this subitem
url	string	1	An API endpoint for getting more information
keywords	list	1	A list of strings describing the content
subitems	list	?	A list of subitems that the subitem may contain

A subitem should contain any additional information that is relevant for the content.

Expanding the results

If the content server is able to provide more attributes, it may always attach them to the response or they can be specifically requested. In this way, the normal response can be short and extra information is available only if it needed.

For example, if the video content server can provide a list of available subtitles, it could be possible to make the following request:
http://our.contentserver.example/api/v1/content/videos/part1?expand=subtitles

By adding the parameter `expand` after the normal request, the requested attribute should be present in the response. Multiple extra attributes can be specified in a comma-separated format.

There is one special attribute that should be supported, `children`. If this is present, the request should contain all the children elements. This means that if this attribute is present when requesting the content types available, the content type response will contain `children` containing all the content packages which have in a same way all the content items inside the package and so on.

Content packages and content items may also support `launch_urls` attribute. If this attribute is present, there will be an additional key-value list of URLs to be used to launch the content by using different launching protocols. An example:

```
"launch_urls":{
   "aplus":" http://our.contentserver.example/aplus/videos/part1",
   "html":" http://our.contentserver.example/videos/part1",
   "pitt":" http://our.contentserver.example/videos/part1?protocol=pitt",
   "lti":" http://our.contentserver.example/lti/aplus/videos/part1"
}
```

The service using these protocols have to know how to use the given launching URL because it may contain some dynamic parameters that are not specified in the response but are still needed to launch the item by using that protocol.

Making queries

Any request can also contain filters to select only some items. For example, only Python videos that are related to variables can be retrieved with the following query:
http://our.contentserver.example/api/v1/content/videos/python?keyword=variables

Server should support at least `keyword` and `author` attributes but more filters may be available. When making queries, the keywords, for example, should propagate to upper levels. If there is keyword in a content item, this keyword should also be defined in the content package level and so on.