

GSOC10 - DSpace REST API

Further development



This is the page describing GSoC 2010 project ended officially in August 2010. The REST API project is continued after official GSoC date. The new address of wiki page is located at <https://wiki.duraspace.org/display/DSPACE/REST+API>. Please bookmark this page as all further development will be described there. Current GSoC page stays here for historical/documentation purposes.

DSpace REST API - Bojan Suzic

Integration, testing, documentation and further development of DSpace REST services for 1.x and 2.0 versions. - Bojan Suzic

| | |
|---------|---|
| 1 | DSpace REST API - Bojan Suzic |
| 2 | Details |
| 2.1 | Project description |
| 3 | Detailed activities |
| 3.1 | REST API Endpoints |
| 3.1.1 | Repository browsing |
| 3.1.1.1 | Optional parameters |
| 3.1.1.2 | Sorting fields: |
| 3.1.1.3 | Controlling results |
| 3.1.1.4 | Response codes |
| 3.1.2 | Authentication/Authorization |
| 3.1.3 | Repository manipulation |
| 3.1.4 | Content searching |
| 3.1.5 | Statistical info |
| 3.1.6 | Relationships interface |
| 3.1.6.1 | Mandatory parameters |
| 3.1.7 | Visitors' suggestions or wishes |
| 3.2 | Integration in the system |
| 3.3 | Documentation tasks |
| 4 | Information for developers |

Details

| | |
|---------------------------|---|
| Project Title: | DSpace REST API |
| Student: | Bojan Suzic, University of Technology Graz |
| Mentors: | Aaron Zeckoski, Mark Diggory |
| Contacting author: | <i>bojan.suzic AT gmail _DOT_ com</i> - using subject line <i>DSpace</i> |
| SCM Location for Project: | http://scm.dspace.org/svn/repo/modules/rest |

Project description

The REST approach promotes simplification and decoupling of software architecture, enabling further scalability, portability, granularity and simplified interaction of software systems and components.

The aim of this project is to provide DSpace with REST capable API and underlying component, which will enable developers and end-users to exploit the advantages of such approach.

Some of uses this module is intended to provide could be, for instance:

- interaction between DSpace systems and/or other repositories
- automation of different activities, e.g. submission of packages
- integrating repositories in process workflows of other applications or systems
- interaction with many kinds of systems or web applications, such as CMS, LMS, LCMS, VLS, AMS etc
- providing of other approaches to UI, such as client based/run UI
- crawling of repositories, exposing information in structural way

This project is continuation of last year's activities, supported by Google as part of [GSoC 2009](#). In the first stage the basic support for REST for DSpace is provided, exposing many parts of DSpace functionality to the clients.

In this year's GSoC the following activities should be primarily addressed:

- integration of existing code in the system
- alignment of REST API with currently available DSpace features/functionality, e.g. adding of new features
- extending of existing code, in order to provide better handling of management and injection functions
- providing more detailed documentation and examples for end users
- testing activities, e.g. cooperation/coordination with other GSoC 10 project [Unit testing](#)
- promotion of DSpace REST interaction (by taking part in integration with other systems)

Detailed activities

In the following sections main activities are elaborated in detail.


REST API Endpoints

In the following section listed are supported endpoints on the application level. **The items marked with dot (in C column) are in phase of implementation, while other items are considered already working.**

Please note that additional tests should be made in order to ensure proper stability of the whole application.

The sorting of the fields / output results is currently partially supported. This part of the application is implemented independently of the endpoints and will be worked on after the most of endpoints are completed.

Naming convention for endpoints

 DSpace 1.x and 2.x are treating the resources on different way. 2.x is more generalized, suggesting the use of RDF-like interrelation notations.

Repository browsing

[Earlier Implementation Description - GSoC09](#)

| C | Verb | URL | Description | Mandatory parameters | Optional parameters | Sorting fields | Response Data | Formats | Response codes |
|---|------|-----------------------------|--|----------------------|--|---|---|-------------|---------------------------|
| | GET | /communities | Returns a list of all communities on the system or return just top level communities. | - | DSPACE:topLevelOnly=true DSPACE:idOnly=false | DSPACE:id DSPACE:name DSPACE:countitems | The list of communities containing DSPACE:respective fields . Response code details: 204 - if there are no communities on the system | json xml | DSPACE:200, 204, 400, 500 |
| | GET | /communities/{id} | Return detailed information about id community. | id | idOnly=false | - | DSPACE:Fields describing community. | json xml | DSPACE:200, 400, 404, 500 |
| | GET | /communities/{id}/{element} | Return a particular data field found in the community id Fields supported (for element): id - entity identifier, internal to the system name - entity name countItems - number of items under community handle - handle of the community (unique persistent resource identifier) type - entity type (object type in the system) collections - collections contained in the community, ordered by id canedit - states user permission on the community (editing) ancestor - ancestors of the community children - subcommunities, ordered by id administrators - group administrators, ordered by id recent - recent items in the community shortDescription - short description copyrightText - copyright text sidebarText - sidebar text introductoryText - introductory text | id | DSPACE:idOnly=false DSPACE:immediateOnly=true | DSPACE:id DSPACE:name DSPACE:countitems | Respective field info | json xml | DSPACE:200, 204, 400, 500 |
| | GET | /communities/{id}/logo | Return a community logo | id | - | - | Contains community logo (bitstream) | binary | DSPACE:200, 400 |
| | GET | /collections | Return a list of all collections in the system. | - | DSPACE:idOnly=false DSPACE:isAuthorized=false | DSPACE:id DSPACE:name DSPACE:countitems | The list of the collections containing DSPACE:respective fields. Response code details: 204 - if there are no communities on the system | json xml | DSPACE:200, 204, 400, 500 |
| | GET | /collections/{id} | Return detailed information about id collection | id | DSPACE:idOnly=false | DSPACE:id DSPACE:name DSPACE:countitems | DSPACE:Fields of the collection entity. | json xml | DSPACE:200, 204, 400, 500 |

| | | | | | | | | |
|-----|-----------------------------|--|-------------|--|---|---|-------------|------------------------------|
| GET | /collections/{id}/{element} | <p>Return a particular data field found in the collection id.</p> <p>Fields supported (for element):</p> <ul style="list-style-type: none"> id - entity identifier, internal to the system name - collection name licence - collection licence items - items contained in collection handle - handle of the collection (unique persistent resource identifier) canedit - states user permission on the collection (edit) communities - communities collection is a part of countItems - number of the items in the collection type - entity type (object type in the system) shortDescription - short description of the collection introText - introductory text for the collection copyrightText - copyright text for the collection sidebarText - sidebar text for the collection provenance - provenance | id | DSPACE: idOnly=false DSPACE: immediateOnly=true | DSPACE:id DSPACE: name DSPACE: countitems | Respective field info | json xml | DSPACE:200, 204, 400, 500 |
| GET | /items | Return a list of the items in the system | - | - | - | The list of the items containing DSPACE:related fields . | | |
| GET | /items/{id} | Return detailed information about an item. | id | - | id name lastmodified submitter | DSPACE:Fields of the item entity. | json xml | 200, 204, 400, 500 |
| GET | /items/{id}/{element} | <p>Return a particular data field found in the item id</p> <p>Fields supported (for element):</p> <ul style="list-style-type: none"> metadata - item metadata submitter - submitter group isArchived - archival status of the item isWithdrawn - states if the item is withdrawn owningCollection - owning collection of the item lastModified - last modified time collections - collections the item appears in communities - communities the item appears in name - name of the item bitstreams - bitstreams related to the item handle - item handle (unique identified) canedit - states can user edit the item id - item id type - element type bundles - bundles related to the item | id, element | - | - | Respective field info | json xml | 200, 204, 400, 500 |
| GET | /bitstream/{id} | Return bitstream object - usually the library item file. | id | - | - | DSPACE:Fields of the bitstream entity. | json, xml | 200, 400, 401, 403, 404, 500 |
| GET | /bitstream/{id}/{element} | <p>Return a particular data field found in bitstream id.</p> <p>Supported fields (for element):</p> <ul style="list-style-type: none"> mimeType - mime type of file bundles - bundles the bitstream is a part of checksum - checksum of the file checksumAlgorithm - checksum algorithm used description - bitstream description formatDescription - file format description sequenceId - sequence id of the file size - size of the file source - source (typically filename with path information) storeNumber - asset store number where the bitstream is stored userFormatDescription - user's format description name - bitstream name handle - unique id of the bitstream id - internal id of the bitstream type - type of the entity (referring to bitstream) | id, element | - | - | Respective field info | json, xml | 200, 400, 401, 403, 404, 500 |
| GET | /bitstream/{id}/receive | Return bitstream | id | - | - | Return bitstream | binary | 200, 400, 401, 403, 404, 500 |
| GET | /groups | Return a list of the groups in the system | - | - | - | The list of the groups containing related DSPACE:fields . | json,xml | 200, 204, 400, 500 |
| GET | /groups/{id} | Return a group object | id | - | - | DSPACE:Fields of the group entity. | json,xml | 200, 204, 400, 500 |

| | | | | | | | | | |
|--|-----|------------------------|--|------------|---|---|--|----------|--------------------|
| | GET | /groups/{id}/{element} | Return a particular data field found in the group entity id. Supported fields (for element): handle - unique id (external) id - internal id of the group isEmpty - is the group empty members - group members (as users) memberGroups - group members (as groups) name - group name type - entity type (referring to group) | id,element | - | - | Respective field info | json,xm1 | 200, 204, 400, 500 |
| | GET | /users | Return a list of the users in the system | - | - | - | The list of the users containing related DSPACE:fields . | json,xm1 | 200,204,400,500 |
| | GET | /users/{id} | Return a user info | id | - | - | DSPACE:Fields of the user entity. | json,xm1 | 200,204,400,500 |
| | GET | /users/{id}/{element} | Return a particular data field found in the user id. Supported fields (for element): email - user's email firstName - first name fullName - full name handle - handle (unique, external) id - internal id of the user language - preferred language lastName - last name name - name netId - network id requireCertificate - requires certificate to login selfRegistered - is user self registered type - type of the object | id,element | - | - | Respective field info | json,xm1 | 200,204,400,500 |

Note: modifier `idOnly` is referred only to first layer of the results. For all other layers (e.g. nested results) only ids are returned in some cases, due to possible loops. Example: for community containing collections, on second level the response contains only ids for some elements where multiple loops may be created (community->has_collection->has_community....). Other data is modified according to `idOnly` flag.

Optional parameters

| Parameter | Description |
|---------------|--|
| topLevelOnly | returns only top level communities |
| idOnly | if true return only the identifiers for the record |
| immediateOnly | return only direct parent community |
| isAuthorized | return only collections user has permission to work on |
| inArchive | return archived items for respective collection |

Sorting fields:

Not completed!



The sorting of the fields / output results is currently partially supported. This part of the application is implemented independently of the endpoints and will be worked on after the most of endpoints are completed.

| Parameter | Description | Ordering supported |
|---------------|--|--------------------------------------|
| id | sort results by entity id | asc ascending desc descending |
| name | sort results by entity name | asc ascending desc descending |
| countItems | sort results by number of items contained | asc ascending desc descending |
| lastmodified | sort results by date of last item modification | asc ascending desc descending |
| submitterName | sort results by submitter name | asc ascending desc descending |
| submitterId | sort results by submitter id | asc ascending desc descending |

Controlling results

| Parameter | Description | Default | Example |
|-----------|--|---|--|
| _start | position of the first entity to return | 0 (first) | _start=5 to list 6th item and onwards |
| _page | page of data to display | 0 (first) | _page=2, to display second page with query results |
| _perpage | number of results to show on each page | 0 (all) | _perpage=10 to display 10 results per page |
| _limit | maximum number of entities to return | 0 (all) | _limit=50 |
| _sort | | the sort order to return entities in should be comma separated list of field names suffix determines ordering suffixes: _asc, _ascending, _desc, _descending ascending default | sort=name _sort=name,email_desc,lastname_desc |

Response codes

| Code | Description |
|------|-----------------------|
| 200 | OK |
| 201 | Created |
| 204 | No content |
| 400 | Bad request |
| 401 | Unauthorized |
| 403 | Forbidden |
| 404 | Not found |
| 405 | Method not allowed |
| 500 | Internal server error |
| 503 | Service unavailable |

Authentication/Authorization

Currently only standard authentication is supported. The authentication data is provided in the request or header body.

Example:

```
/rest/communities.json?user=user@email.com&pass=userpassword
```

The same elements `user` and `pass` are used for header based authentication.

Authorization is done on underlying api level; in the case of error the proper message and error code are returned to the user.

Repository manipulation

| C | Verb | URL | Description | Mandatory parameters | Optional parameters | Response Data | Formats | Response codes |
|---|--------|-----------------------------|---|---|---------------------|--|-------------|-------------------------|
| | POST | /communities | Action to be done under community id, adding new content or values. Supported actions: createAdministrators createCollection createSubcommunity | id, action - name name | - | Id of newly created entity, depending on the action selected: id of group of administrators id of collection id of subcommunity | json xml | 200, 400, 401, 403, 500 |
| | PUT | /communities/{id}/{element} | Update the field element of the community id. Supported fields: name - change name shortDescription - change short description copyrightText - change copyright text sidebarText - change sidebar text introductoryText - change introductory text collections - add existing collection children - add existing subcommunity | id value value value value value cid cid | - | Response code | | 200, 400, 401, 403, 500 |
| | PUT | /communities/{id}/logo | Set the logo for community id | id | - | Response code | binary | 200, 400, 401, 403, 500 |
| • | DELETE | /communities/{id} | Delete community from the system | id | - | Response code | json xml | 200, 400, 401, 403, 500 |

| | | | | | | | | |
|---|--------|-----------------------------------|---|---|---|-----------------------------|------------------|-------------------------|
| • | DELETE | /communities/{id}/{element}/{eid} | Remove attribute/value eid of element element from the community id. Supported elements: collections children administrators | id, eid cid cid - | - | Response code | json xml | |
| | POST | /collections | Action to be done under collection id, adding new content or values. Supported actions: createAdministrators createSubmitters createTemplateItem createWorkflowGroup | id, action - - - step | - | Id ow newly created element | json xml | 200, 400, 401, 403, 500 |
| | PUT | /collections/{id}/{element} | Update field element of the collection id. Supported elements: shortDescription - short description introText - introductory text copyrightText - copyright text sidebarText - sidebar text provenance - provenance licence - collection licence name - collection name | id value value value value value value value | - | Response code | json xml | 200, 400, 401, 403, 500 |
| • | DELETE | /collections/{id} | Delete collection from the system | - | - | Response code | json xml | 200, 400, 401, 403, 500 |
| • | DELETE | /collections/{id}/{element}/{cid} | Remove attribute/value cid from collection id Supported attributes: administrators item submitters templateItem | id, cid | - | Response code | json xml | 200,400,401,403,500 |
| • | PUT | /collections/{id}/logo | Set the logo for collection id | id | | Response code | binary | 200,400,401,403,500 |
| • | POST | /items | Action to be done under item id, adding content or value. Supported actions: createBundle createBitstream | id, action name name, input | | Id of newly created element | json,xml, binary | 200,400,401,403,500 |
| • | PUT | /items/{id}/{element} | Update field element of the item id Supported fields: isArchived isWithdrawn owningCollection submitter | id,element | - | Response code | json,xml | 200,400,401,403,500 |
| • | DELETE | /items/{id} | Delete item from the system | id | - | Response code | json,xml | 200,400,401,403,500 |
| • | DELETE | /items/{id}/{element}/{eid} | Delete element/attribute eid from the item id Supported fields for element: bundle licences | id,element,eid | - | Response code | json,xml | 200,400,401,403,500 |

Content searching

| C | Verb | URL | Description | Mandatory parameters | Optional parameters | Sorting fields | Response Data | Formats | Response codes |
|---|------|----------|---|----------------------|---|---|--|-------------|--------------------|
| | GET | /search | Return a list of all objects found by searching criteria. Notice: community and collection are mutually exclusive options. | - | modifiers({query=query})& (community=id or collection={{id}}) idOnly=false | id name lastmodified submitter | Item info with basic metadata for the search results. Additionally return only identifiers when idOnly=true is used. | json xml | 200, 204, 400, 500 |
| | GET | /harvest | Return a list of all objects that have been created, modified or withdrawn within specified time range. | - | startdate {enddate} community collection idOnly=false withdrawn=false Notice: community and collection are mutually exclusive options | - | Contains item info including id, name, handle, metadata, bitstreams according to the defined requirements. Additionally when idOnly=true only identifiers of results are returned. If the date is in incompatible format, error 400 is returned. | json xml | 200, 204, 400, 500 |

Statistical info

| C | Verb | URL | Description | Mandatory parameters | Optional parameters | Sorting fields | Response Data | Formats | Response codes |
|---|------|--------|----------------------------|----------------------|---------------------|----------------|---|-------------|----------------|
| | GET | /stats | Return general statistics. | - | - | - | Cummulative list of statistics data for the system currently available. | json xml | 200, 400, 500 |

Relationships interface

Experimental feature

This is considered as an experimental feature in the phase of being considered for compatibility with future versions of DSpace. Consider not important section; the status of the feature for upcoming release yet to be determined.

| C | Verb | URL | Description | Mandatory parameters | Optional parameters | Sorting fields | Response Data | Formats | Response codes |
|---|------|------------------------------|--|---------------------------|------------------------|----------------|---|-------------|-------------------------------------|
| • | GET | /resource/{handle}/relations | Return entities according to relation and parameters specified | handle DSpace:property | DSpace:rtype rfield | - | contains entities selected and sorted in conformance to request parameters. For more details see description of rtype and rfield. | json xml | DSpace:200, 204, 400, 401, 403, 500 |

Mandatory parameters

| Parameter | Description | Values | Example |
|-----------|---|---|---|
| property | Return entities satisfying requested property relation | Structural properties ds:isPartOfSite ds:isPartOfCommunity ds:isPartOfCollection ds:isPartOfItem ds:isPartOfBundle ds:hasCommunity ds:hasCollection ds:hasItem ds:hasBundle ds:hasBitstream ds:hasBitstreamFormat Communities and collections ds:logo Bistream format ds:support ds:fileExtension ds:mimeType Bitstream ds:messageDigest ds:messageDigestAlgorithm ds:messageDigestOriginator ds:size Eperson ds:language | property=ds:hasCommunity - return subcommunities of a community property=ds:isPartOfCommunity - return communities current community is part of (children) property=ds:hasCollection - return collections belonging to community property=ds:hasItem - return Items belonging to community |
| rtype | restriction on type - only entity with specified type(s) would be returned | ds:Bitstream ds:Bundle ds:Collection ds:Community ds:EPerson ds:Group ds:Item ds:DSpaceObject ds:Policy ds:Site ds:BitstreamFormat | rtype=ds:Collection - return entities of Collection type |
| rfield | restriction on fields - return only selected fields; by default all fields are returned | id name countitems metadata subcommunities ancestors owner other (depending on object type, will be documented later) | rfield=id,name - return only entity id and name in response |

Note: incomplete/orientative properties, for more info check [\[Vocabularies\]](http://code.google.com/p/dspace-sandbox/source/browse/#svn/modules/dspace-rdf/tags/dspace-rdf-1.5.1/src/main/java/org/dspace/adapters/rdf/vocabularies) <http://code.google.com/p/dspace-sandbox/source/browse/#svn/modules/dspace-rdf/tags/dspace-rdf-1.5.1/src/main/java/org/dspace/adapters/rdf/vocabularies>.

Visitors' suggestions or wishes

Here the visitors and stakeholders can insert their suggestions or describe the needs for their applications in detail.

Comment: In this case it is not clear how to treat recent part of endpoint. If we stick to semantic mapping, then it should look like /resource/id/mapping, but recent in this case obviously do not represent a mapping, but the property.

Comment #2: Semantic mapping presented in this case should be probably hardcoded for 1.x branch, but on abstraction level which enables easy replacement with some auto-discovery method prepared for 2.x and eventually backported to 1.x. This way we would be able to call something similar to /communities/id or communities/id/capabilities in order to get supported mappings (amongst other data).

Suggesting new options:

Instead of changing wiki contents visitors can enter their suggestions as a comments.

1) **Kevin S. Clarke and Tim Donhue** suggested adding of the new feature related to HTTP Basic Auth. Ok, I will investigate how it could be done and included here. More info coming.

Integration in the system

It is planned to consult two external subjects for cooperation and the assistance during integration process (LMS and national library internal automation process). More information coming soon - awaiting approval of other parties.

Documentation tasks

Although provided software module exposes basic documentation automatically to the end user, in order to make it easier for other developers and users the documentation in the following forms is additionally to be provided:

- Confluence pages, current location
- integrated documentation in PDF form (manual)
- short slides containing technology overview, advocacy/facts, configuration and usage guidelines and examples
- code will be additionally commented

Example of usage

At the end of the current stage of this project as a bonus task (if time constraints allow) the examples of usage will be provided for several languages, the use-cases will be presented (example of integration in other software, e.g. LMS) and optionally simple client system demonstrating UI customization will be demonstrated (e.g. Flex or JavaFX like).

Information for developers

In this section the main sections of the software will be briefly explained in order to ease update or extension of the components.

The REST API for DSpace uses Aaron Zeckoski's [EntityBus](#) and DSpace standard libraries, as dspace-api.

There are two main packages: **org.dspace.rest.providers** and **org.dspace.rest.entities**. Providers are responsible for serving content/feeds to the users. They usually prepare entities or particular entity and/or handle update/delete/create functions. The main class there is **AbstractBaseProvider**, which is extended by other providers.

In the providers, at the constructor level created are mappings between particular endpoints (e.g. /rest/items/1/collections) and related functions in entities (org.dspace.entities.items.getCollections). Thus, for each GET, PUT, POST or DELETE function in the entity provider's constructor defined are such mappings between URL endpoints and functions. After the client makes specific call, the provider prepares answer and in this phase calls mapped function and prepare results for display.

So, if you want to extend currently available endpoints for already present providers and entities, it is necessary to define mapping at provider level and prepare corresponding function at the entity level (based on the template). The system then calls this function and provides necessary arguments for its successfully handling.

If you want to develop a new provider, it is usually necessary to create new provider class in org.dspace.rest.providers and then create related entity in org.dspace.rest.entities. The currently available providers are good example how to do that.