DOI support using DataCite

This site is outdated!

This site is outdated. Since DOI support was inculded in DSpace, the official DSpace documentation contains a site about DOIs. Please use the documentation of your DSpace version!

Code can be found in our DSpace repository on GitHub, in the DOI branch: https://github.com/tuub/DSpace/tree/DOI.

DOI support

We (at Technische Universität Berlin) want to use DOIs for Items within DSpace. We are thinking about using DOIs for Communities and Collections, but at first we'll concentrate on items. DOI is a well known persistent identifier. With the external identifier support @mire introduced to DSpace 3.0 with the item versioning feature it should be possible to add support to mint, register and delete DOIs using DSpace.

Registration agencies, DataCite and EZID

To register a DOI one has to enter into a contract with a DOI registration agency. Several such agencies exist. Different DOI registration agencies have different policies. Some of them offer DOI registration specially or only for academic environment, others only for publishing companies. Most of the registration agencies charge fees for registering DOIs, all of them have different rules describing for what kind of item a DOI can be registered. To implement DOI support for DataCite we have to be mindful of the fact that every registration agency has their own API (see below).

DataCite is an organization that aims to support the access to, the acceptance of and the archiving of research data. One of the services offered by DataCite members is DOI registration. DataCite has several members that act as a DOI registration agency. Some of the members tell their customers to use the API of DataCite directly, others offer their own APIs. **So to register a DOI at a member of DataCite does not automatically mean to use DataCites API directly.**

We will register our DOIs using the service of TIB Hannover, a German member of DataCite. We will use the DataCite API directly. EZID is a DOI registration agency in the USA that is also part of DataCite. EZID offers their own API, so that EZID customers won't profit directly from our development.

DOIIdentifierProvider, DOIConnector and DataCiteConnector

Knowing this situation we developed a DOIIdentifierProvider that should perform everything that is necessary to support DOIs on the DSpace side. For example, after minting and registering a DOI it saves the DOI as a metadata value of an Item. To be able to extend our DOIIdentifierProvider, we put a DOIConnector between our DOIIdentifierProvider and the registration agency API. The DOIConnector has to implement seven methods and should be quite easy to implement for any API of a DOI registration agency. The seven methods are:

- a method to check if a DOI is already reserved,
- a method to check if a DOI is reserved for a given DSO,
- a method to check if a DOI is already registered,
- a method to check if a DOI is registered for a given DSO,
- a method to reserve a DOI for a given DSO,
- a method to register a DOI for a given DSO,
- a method to delete a DOI for a given DSO.

We already developed a DataCiteConnector that implements these methods for everyone who uses the DataCite API directly. As told above, EZID has their own API, but it should be quite simple to implement a DOIConnector providing these seven methods with the EZID API.

Metadata

DataCite wants to get metadata of the objects the DOIs addresses. The DataCite Schema (http://schema.datacite.org) defines an XML structure to describe the metadata of an object. We developed a DIM2DataCite crosswalk that takes the metadata of a DSpace Item and transforms it into a XML using DataCite Schema 2.2. As far as I know, EZID does not use this XML so another crosswalk is probably needed. It should be discussed (see below or in the JIRA ticket) how we want to deal with metadata updates, as the API for external identifiers does not define a mechanism to update metadata for an external identifier yet.

DOIOrganiser

While developping DOI support for DSpace we recognized that DSpace would become dependent on the registration agencies API. If the API is offline or malfunctioning for any reason DSpace wouldn't be able to receive new submissions. To resolve this dependency we introduced DOIOrganiser. Instead of reserving and registering DOIs at the moment DSpace asks to, we just write into the database that a DOI should be reserved or registered. A cronjob checks periodically if there are DOIs to reserve or register. If the registration agencies API is offline, DSpace can still get new submissions. In case of an error, the cronjob will retry to reserve or register a DOI when it runs the next time.

The downside of this design decision is that there's a delay in reservation and registration of DOIs. However the duration of the delay depends on how often you run the cronjob. The big advantage is that DSpace stays independent from the status of external services (in this case the registration agencies API). For further information how to use the DOIOrganiser see the online help ([dspace-install]/bin/dspace doi-organiser --help).

A DOI will be registered in the item metadata after online registration. If you activate DOI support within DSpace every new Item gets an DOI. The DOI will be added to item metadata when the DOIOrganiser was able to register it at the registration agency.

How to Test

Of course you first have to get our code and compile it on your test system. You can find the code on github (https://github.com/tuub/DSpace/tree/DOI). If you use this code on an existing DSpace test installation, you have to add the DOI table to your database. Please make sure you have a backup first. Please dump your database! You can use either https://github.com/tuub/DSpace/blob/DOI/dspace/etc/postgres/database_schema_3-4.sql or https://github.com/tuub/DSpace/blob/DOI/dspace/etc/oracle/database_schema_3-4.sql depending on the database system you use.

To register DOIs you will need an account from a registration agency that allows you to use the DataCite API directly (EZID provides an API on their own). TIB Hannover was so friendly to provide us a test account for DSpace development. At DataCite every account has to define URLs it want to be able to register DOIs for. With the following test account you can register only DOIs that point to example.org or any subdomain of it. So you have to configure your computer to resolve example.org to your test installation and you have to configure your test installation as if it would run under example.org (this is done by setting dspace.hostname and dspace.baseUrl on appropriate values). The username of the test account is "TIB.DSPACE", the password is "duraspace" and the prefix is 10.0128. You can reach the web user interface of the test system under https://test.datacite.org where you can see which DOIs got reserved and registered. All DOIs registered in the test system will be deleted from time to time.

DOI support is disabled by default. So even with our code your have to configure DSpace to use it. In dspace.cfg you'll have to configure the properties identifier.doi.user, identifier.doi.password, identifier.doi.prefix and identifier.doi.namespaceseparator. The namespaceseparator will be prefixed to every suffix DSpace generates. If you use 10.0128 as prefix, "abc-" as namespaceseparator the DOIs generated by DSpace will look like 10.0128/abc-1, 10.0128/abc-2, ..., 10.0128/abc-1024, ... The namespace separator is usefull if different software or multiple DSpace installations use the same prefix. For testing please use something you think it's unique as namespace separator (f.e. your name, your login, ...). In dspace.cfg you have to remove the comments from event.consumer.doi class and event.consumer.doi.filters and add doi to event.dispatcher.default.consumers. The second file you have to configure is dspace/config/spring/api/identifier-service.xml. In this file you have to remove to comments around the bean for the DOIIdentifier and around the bean for the DataCiteConnector. The properties for the DataCiteConnector has to be set as following:

```
 <property name='DATACITE_SCHEME' value='https'/> <property name='DATACITE_HOST' value='test.datacite.org'/>
<property name='DATACITE_DOI_PATH' value='/mds/doi/' /> <property name='DATACITE_METADATA_PATH' value='/mds
/metadata/' /> <property name='disseminationCrosswalkName' value="DataCite" />
```

The last change you should make is in config/crosswalks/DIM2DataCite.xsl. You should add the name of your institutions there. DIM2DataCite.xsl gets used to convert metadata before they will be send to DataCite. The file contains a configuration section right at the beginning, you should understand it without further help (just see the comments in DIM2DataCite.xsl).

When you make new submissions they should get DOIs now. You should be able to see DOIs as part of the metadata of an item. Remember to run the DOIOrganiser (see above) to reserve and register DOIs at the DOI registration agency.

Status

On 2013/09/27 we pushed a new version of our code to github (https://github.com/tuub/DSpace/tree/DOI). We rebased the code onto DSpace master. So if you downloaded our code before 13/09/27 you probably will get problems if you just try to download the new code by using git pull.

As with every code there is always something left that could be improved. But we think our code is ready to get tested and inclu uld ussions should happen in the JIRA ticket (Unable to locate Jira server for this macro. It may be due to Application Link configuration.). In the comments to this ticket you can s our toc

Design Decisions

There are two design decisions we want to document here:

We safe the DOIs in the database using a column type VARCHAR(256). As far as I know there is no limit for the length of a DOI. We decided to use a VARCHAR(256) column as it provide quite a good length and is fully indexed. This is important as DOIs must be unique and we want to use support by the database to find a DspaceObject by its DOI really fast.

When we mint a new DOI it is generated by the database. We do not check if it already exists as the database should generate one DOI after the other without any conflict. If one adds DOIs manually to the database it's on his or her own risk. Nobody should insert DOIs without updating the database seqeunece doi_seq. Nevertheless it is possible to use a different name space separator when one adds DOIs manually. Almost the same applies to the registration agency: We check if a DOI is reserved or registered online but we don't have any strategy to handle conflicts. So please use a unique name space separator that only one DSpace installation use. In other words: If you want to avoid trouble, DSpace needs a space where only it reserves and register DOIs.

Thanks

I want to thank Mark H. Wood who made the first steps for a DOIIdentifier. His code helped me to understand how DSpace handles metadata and what should be done to support DOIs within DSpace. He started to implement a DOIIdentifierProvider using EZID and also wrote a test class for it. I took the liberty to use some of his code.

The other persons helping me were Fabian Fürste and Marsa Haoua, colleagues here at TU Berlin. Thanks goes to them as well.

I also want to thank TIB Hannover for providing a test account to the DSpace community.