OSUG-DOI (public)

Source code: https://gricad-gitlab.univ-grenoble-alpes.fr/OSUG/DC/osug-doi

Roadmap:

- tests: metadata + soumission DOI de test: OK
- génération landing pages simples: OK
- génération metadata et landing pages pour amma-catch (pipeline iso 19-115 + templates csv): OK
- TODO List:
    - TEST:
    1. publication DOI avec relatedIdentifier en test (lien invalide): fait = datacite ne vérifie pas la validité des relations
    - DOC:
    1. faire un template CSV (with all possible key/values + comments)
    2. procédure à écrire pour un projet (templates, config...)
    - DEV:
    1. création dépot DOI sur forge (sourcesup ?): OK
    2. packaging scripts, pipeline + configuration (templates), landing pages: OK
    3. améliorer landing page (html header / footer, about page): OK
    4. générer index.html pour lister les DOIs (staging / public): en cours
    5. gestion BDD:
        1. design des tables: projet et DOI: dates (génération, publication), état(staging/public), URL data access, URL landing page externe, URL landing page active: en cours
    6. améliorer scripts de publication pour gérer des dossiers staging et production (par projet) et modifier l'etat en BDD: en cours
    7. accès landing pages: gestion des redirections si landing page personnalisée (générer .htaccess)
    8. supervision pour detecter / signaler les URLs invalides (URL 404)
    - PROD:
    1. création vm doi.osug.fr pour héberger toute la gestion (backoffice simple) et les landing pages
        1. demande: faite
        2. installation paquets (httpd + conf)
        3. installation pipeline + scripts
        4. installation BDD
- backoffice web de gestion/soumission DOI (futur)

Principe

Roles:

- Datacite: assure la résolution du DOI et possède les informations pour chaque DOI: méta données (xml) + URL associée vers doi.osug.fr (landing page ou ressource)

- OSUG-DC: le serveur doi.osug.fr héberge les landing pages (en local) ou réalise une redirection vers un autre serveur (spip...) dans le périmètre de l'OSUG
- les landing pages doivent être pérennes et contenir les instructions pour accéder aux données / services

Développement

OSUG DOI: http://doi.osug.fr/

Génération Landing Pages (datacite XML to HTML)

A partir d'un document XML correspondant aux métadonnées du DOI (datamodel datacite), une transformation XSLT génère une page HTML statique (bootstrap + CSS) qui affiche simplement les méta données.

Cependant, il manque les informations pour l'accès aux service / données dans le document XML (datacite).

TODO:

- trier les meta données selon le document AMMA-CATCH: OK
- ajouter un page footer (date + lien vers OSUG-DOI): OK
- rajouter les instructions d'accès aux données via un fragment HTML (fichier dans config); si manquant, prévoir une phrase par défaut
- rajouter l'URL de destination pour accéder aux données
- revoir le page header (optionnel)
- générer l'index pour staging / public

Proposition: fournir ces paramètres à la transformation XSLT récupérés depuis la configuration (projet ou du jeu de données pour URL) depuis les templates CSV (data_access:URL et data_access:information) ou la BDD ?

Outil de génération des métadonnées DOI (datacite XML) à partir des méta données ISO-19139 (CSW)

Cas d'utilisation = AMMA-CATCH (voir http://bd.amma-catch.org/amma-catch2)

Principe

Certaines infos nécessaires au schéma datacite sont déjà renseignées pour chaque jeu sur le portail Web d'accès aux données AMMA-CATCH et sont récupérables par http en utilisant le webservice CSW méthode getRecords (format XML ISO 19139):
http://bd.amma-catch.org/amma-catchWS2/WS/csw/default?service=CSW&request=GetRecords&version=2.0.2&typenames=csw:Record&resulttype=results&maxrecords=100&elementsetname=summar

y&outputschema=http://www.isotc211.org/2005/gmd

Il faut ensuite les compléter pour produire un fichier XML DataCite pour chaque jeu de données.

Chaine de fusion des informations (CSW + templates)

Pour compléter les metadonnées, des templates (global, pays et jeu) au format CSV sont utilisés.

Séquence de traitement:

1. GetRecords → renvoie autant d'élements <gmd:MD_Metadata></gmd:MD_Metadata> qu'il y a de jeux de données à condition d'indiquer le paramètre maxrecords dans l'URL
2. Conversion au format CSV
3. Pour chaque jeu de données, fusionner avec les templates:
    1. template global: template_all.csv
    2. template pays: correspondant au pays indiqué par geoLocationPlace (geoLocationPlace;Benin)
    3. template spécifique au jeu de données: correspondant à l'identifiant indiqué par identifier (identifier:DOI;10.5072/AMMA-CATCH.CL.Run_O)
    4. sauvegarder et convertir au format XML datacite
4. validation des fichiers XML avec le schéma
5. vérifier les related identifiers (existe en BDD ?) pour tester des références invalides ?
6. générer les landing pages associées
7. tester publication du DOI (metadata + URL) avec le préfixe de test: STAGING
8. sauvegarder les informations du DOI en BDD (statut des étapes de traitement, URL ...)

Note: les identifiants DOI doivent contenir que le préfixe de test (publication ultérieure) et les méta-données en doublons sont ignorées (même clé / valeur) mais attention si les informations ne sont pas disjointes (clés présentes dans plusieurs templates)

Attention à l'encodage des fichiers CSV (utf-8 sous linux et iso-8859-1 sous windows) !

TODO:

- généraliser à d'autres services CSW en externalisant toute la configuration du pipeline dans un fichier (clé=valeur) par projet: en cours
- validation: vérifier la présence de méta-données (en plus, de la validation xml assez laxiste) comme dataAccess:URL ...
- comment gérer la mise à jour des templates CSV ? manuel dans un premier temps; plus tard: via owncloud.osug.fr ou via interface d'administation

Gestion multi-projet dans le file system:

- config
    - AMMA-CATCH
    - projetX
- staging
    - AMMA-CATCH
    - projetX

- public
  - AMMA-CATCH

Publication DOI (datacite)

objectif: automatiser la publication des DOIs et alimenter la BDD (statut)

TODO:

1. copier la landing page (URL publiée chez datacite): PUBLIC
2. reécrire les identifiants DOI (identifier et relatedIdentifier) pour remplacer le préfixe de test en préfixe OSUG à l'aide de la base de données
3. publier ce DOI avec le préfixe OSUG et mettre à jour la BDD
4. gérer les redirections ie une URL alternative pour la landing page

note: garder les documents XML + URL dans le file system et BDD