

HEG-796-22-020

LDP en pratique

Jan Krause-Bilvin

2022-04-25

Thème de cette session

-
- Plateforme LDP [Fedora Commons](#)
 - Description archivistique [Records in Context](#) et l'ontologie [RiC-O](#)
-

Cours précédent

-
- Linked Data Platform (LDP):
 - Ressources (`ldp:Ressource`) de type RDF et non-RDF
 - Conteneurs (`ldp:Container`), peuvent être emboîtés.
 - Manipulation via verbes HTTP (GET, POST, PUT, DELETE).
 - Les conteneurs LDP permettent de délimiter les ressources représentant des objets (métier, archivistiques).
 - OAIS: objet archivistique = Archival Information Package (AIP).
-

Records in Context

RiC existe sous deux formes:

- Le modèle conceptuel: RiC-CM
- L'ontologie OWL: RiC-O

RiC-O est une implémentation possible de RiC-CM, pour l'instant la seule. Mais d'autres implémentations sont envisageables (cf. ébauche: [ici](#) et [ici](#)).

- RiC-CM est un modèle entité relation.
- Toute entité est une chose (Thing) et se décline en différentes classes.
- Les entités sont liées par des relations.

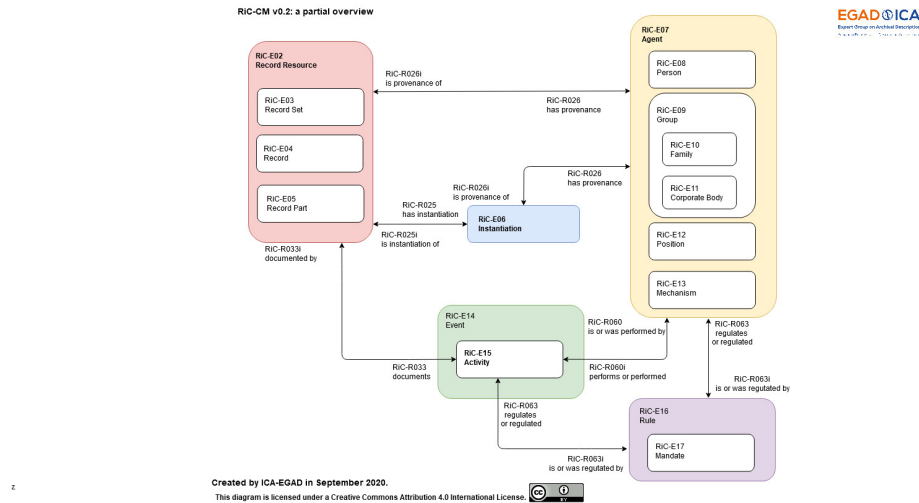


FIGURE 1 – RiC overview

RiC-O : principes

- Ontologies de référence/domaine de l'archivage
- Utilisable immédiatement
- Flexible (granularité variable)
- Nouveaux potentiels (interprétable/SPARQL, instantiations)
- Extensible (autre contextes que les archives, combinaison)

Des concepts complétant RiC-CM ont été développés pour créer RiC-O, p.ex.:

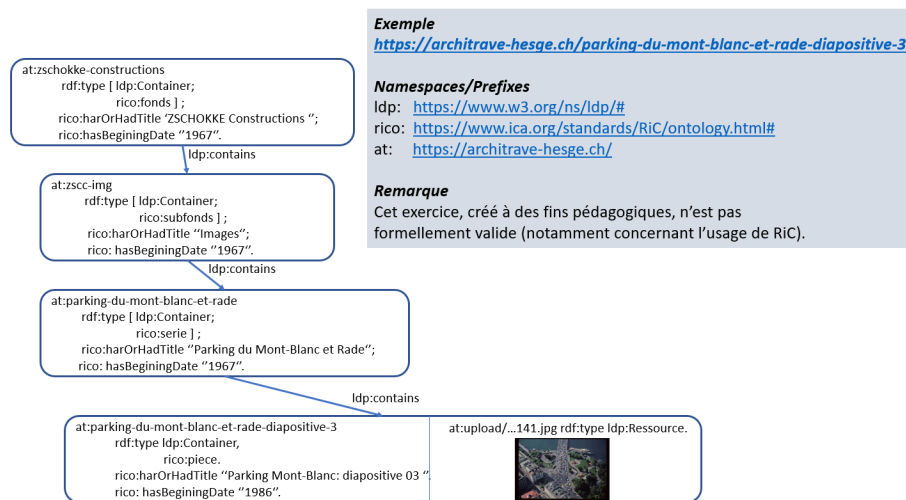
- **rico>Type** : gestion de types d'entités, utiles pour caractériser les entités et se lier à d'autres ontologies/vocabulaires comme **W3C-SKOS**.
- **rico:Proxy** : intégrer un record dans plusieurs record sets (ex: un document élaboré par deux services).
- **rico:Place** : les lieux peuvent évoluer au cours du temps (ex: frontières qui se déplacent, sur plus de 500 ans dans le cas de la Suisse), voir aussi **linked-places**.

Concepts clés de RiC-O (centrés sur les records):

Url de l'ontologie. Quelques exemples:

- Intitulé, titre: **rico:title**
- Créateur: **rico:hasCreator**
- Type: **rico:hasRecordSetType**
- Hierarchie: **rico:hasOrHadPart** , **rico:isOrWasPartOf**
- État: **rico:hasRecordState**
- Date: **rico:hasBeginningDate** , **rico:hasEndDate**

Correctif exercice TP



Geslon des containers via l'interface web

Créer un container (interface Web)

The screenshot shows the Fedora Web interface. At the top is a navigation bar with 'Fedora', 'Home', 'Transactions', and 'Search'. Below this, the URL 'http://localhost:8080/rest/' is displayed. A 'Home' button is visible. The main content area is divided into two columns. The left column shows metadata for a resource: 'Created at' (2022-03-23T08:35:41.9771551Z by), 'Last Modified at' (2022-03-23T08:35:41.9771551Z by), and 'Children' (0). Below this is a 'Properties' tab with a '+' icon. The right column is titled 'Create New Child Resource'. It has a 'Type' dropdown menu set to 'basic container'. Below that is an 'Identifier' field with the text '(auto-generated identifier)'. At the bottom of this column is an 'RDF (turtle)' section with a text area containing the following code:

```
<>
<https://www.ica.org/standards/RiC/RiC-0_v0-2.html#title>
  "Le titre du record".
```

 Below the text area is an 'Add' button.

Dans la boîte de dialogue “RDF Turtle”, insérer:

```
<>
<https://www.ica.org/standards/RiC/RiC-0_v0-2.html#title>
"Le titre du record".
```

Cliquer ensuite sur le bouton “Add” comme dans la capture d’écran ci-avant.

A noter que le container est créé comme enfant du container courant.

Modifier un container (Web)

The screenshot shows a web interface for modifying a container. On the left is a metadata form with fields for 'lastModified' (2022-03-23T15:09:07.834161600Z), 'title' (Le titre du record), and 'rdf:type' (a list of URIs including http://fedora.info/definitions/v4/repository#Container). Below this is a section for 'Other Resources'. On the top right is an 'RDF (turtle)' editor showing a single triple: a <http://purl.org/dc/dcmitype/Collection>. Below this is an 'Add' button. On the bottom right is an 'Update Properties' dialog box. This dialog contains a text area with the following content:
~~DELETE{~~ <> <https://www.ica.org/standards/RiC/RiC-O_v0-2.html#title> "Le titre du record". }
INSERT{ <> <https://www.ica.org/standards/RiC/RiC-O_v0-2.html#title> "Le nouveau titre du record". }
Below the text area is an 'Update' button.

Dans la boîte de dialogue “Update Properties”, remplacer:

```
DELETE { }  
INSERT { }
```

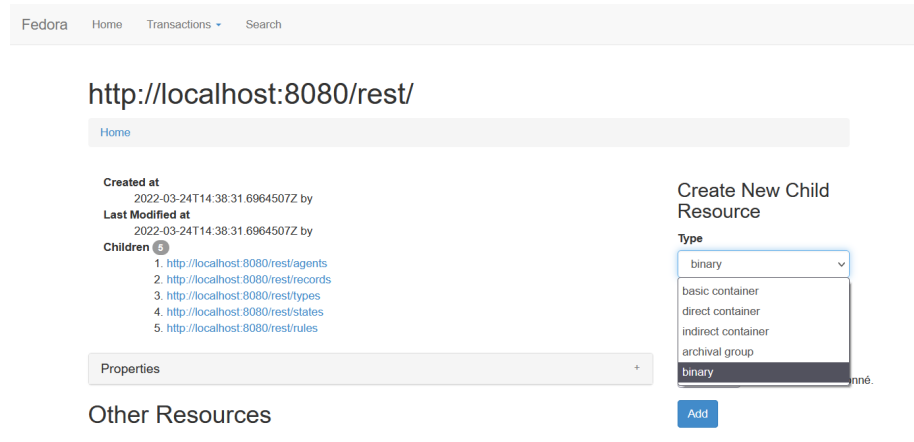
Par

```
DELETE{ <> <https://www.ica.org/standards/RiC/RiC-O_v0-2.html#title> "Le titre du record".  
INSERT{ <> <https://www.ica.org/standards/RiC/RiC-O_v0-2.html#title> "Le nouveau titre du r
```

Puis cliquer sur le bouton “Update”.

Cf. capture d’écran ci-avant.

Créer une ressource binaire



Dans la section “Create New Child Ressource”, choisir “Type” : “binary” dans la liste déroulante comme dans la capture d’écran précédente.

Puis cliquer sur “Parcourir” (ou “Browse”) pour sélectionner le fichier à envoyer dans Fedora.

Valider en cliquant sur le bouton “Add”.

Gestion des containers par HTTP

Les verbes standard standard sont utilisés (**API rest**):

- Accéder : GET
- Créer / Mettre à jour : POST / PUT
- Supprimer : DELETE

Rappel: python

Dans le contexte de la HEG, python est installé sur les postes via l’outil Anaconda.

Pour l’exécuter, ouvrir:

Windows > Menu démarrer > Anaconda prompt

Puis taper:

```
python
et enter.
```

Python sur ordinateur personnel

Dans le cadre d'une installation classique de Python 3 ([téléchargement](#)), installer le paquet "requests" si ce n'est pas déjà fait:

Sous Linux:

```
pip3 install requests
```

ou

```
pip install requests
```

Sous Windows:

```
python -m pip install requests
```

Sous MacOS:

```
sudo easy_install pip
sudo pip install --upgrade pip
```

Accéder à une ressource

```
import requests
url = 'http://localhost:8080/rest/records/acv/D000002513'
r = requests.get(url)
print('Status code:', r.status_code)
print(r.text)
```

Créer un container

```
import requests
url = 'http://localhost:8080/rest/records/acv/D9999'
headers = {"Content-Type": "text/turtle"}
auth = ('fedoraAdmin', 'fedoraAdmin')
data = """ <> <rico:title> 'Ceci est le titre'.
          <> <rico:scopeAndContent> 'Voilà la description'.
          """
r = requests.put(url, auth=auth, data=data.encode('utf-8'), headers=headers)
print('Status:', r.status_code)
print(r.text)
```

Mettre à jour un container

```
import requests
url = 'http://localhost:8080/rest/records/acv/D9999'
headers = {"Content-Type": "text/turtle"}
auth = ('fedoraAdmin', 'fedoraAdmin')
data = """ <> <rico:title> 'Ceci est le titre mis-à-jour.'.
          <> <rico:scopeAndContent> 'Et la description revue'.
          """
r = requests.put(url, auth=auth, data=data.encode('utf-8'), headers=headers)
print( 'Status:', r.status_code )
print( r.text )
```

Voir les section Versionning : “View Versions”

Créer une ressource binaire

Le code suivant suppose qu’une ressource binaire nommée “image.jpg” se trouve dans le répertoire “Pictures” (le répertoire d’images par défaut sous Windows) et que Python a été lancé depuis votre répertoire personnel racine.

```
import requests
import os
filename = 'Pictures' + os.sep + 'image.jpg'
mimetype = 'image/jpeg'
data = open(filename, 'rb').read()
url = 'http://localhost:8080/rest/records/acv/D9999/binary'
auth = ('fedoraAdmin', 'fedoraAdmin')
headers = { "Content-Type": mimetype,
            "Link" : "<http://www.w3.org/ns/ldp#NonRDFSSource>; rel=type"}
r = requests.put(url, auth=auth, data=data, headers=headers)
print( 'Status:', r.status_code )
print( r.text )
```

Versionning RFC 7089 (Memento)

Memento est le protocole de navigation temporel du Web.

— Par défaut, Fedora Commons conserve toutes les versions des ressources.

- Il est possible de personnaliser ce fonctionnement.
- Cela permet de naviguer dans le temps (p.ex. mise à jour du plan de classement).

Voir aussi: *Memento at W3C* Timetravel, avec [cet exemple](#) et [celui-ci](#).
