

Installation guide of DSpace plug-in for creating the Europeana schema in a DSpace repository environment, execution and configuration

Kostas Stamatis

(kstamatis@ekt.gr)

National Documentation Center

March 23, 2010

The purpose of this document is to describe the function, the installation procedure, the execution and the configuration of the plug-in that has been developed in *National Documentation Center* for the addition of the **Europeana** schema in any *DSpace* repository and the automated completion of the metadata fields. The specified function is proposed to be merged to *DSpace plug-in for Europeana Semantic Elements* which has been developed by Evaggelos Banos (<http://vbanos.gr/?p=189>) and is used for the collection of the Greek content that is made by the Public Library of Veroia in the context of the **EuropeanaLocal** project. The specified function can be an alternative solution for the automated update that is provided with the application *metadata_updater.php* with the following characteristics:

- Automated creation of the Europeana schema in the *DSpace* repository – no need for manual creation of the schema through the administration environment of *DSpace*.
- Configuration capabilities through the *DSpace* configuration file named *dspace.cfg* without need of changing the *plug-in* code.
- Use of *DSpace API* for metadata value updates – no direct connection to the database
- Execution with no other prerequisites (like *PHP*) than a *DSpace* installation

Introduction

DSpace, in its default installation includes the *DC (Dublin Core)* schema. However, in order for the repository items to be used in *Europeana*, these items must include metadata according to *Europeana* schema. The plug-in that has been developed in *EKT* has the capability of creating the new schema for the user (in case of no schema existence in the repository) and fills automatically the metadata values using the values from the *DC* metadata fields. This plug-in can be configured using the *DSpace* configuration file (*dspace.cfg*) in order to meet the needs of any repository without changing a line of code.

The plug-in has been developed using the Java programming language and utilizes the *DSpace API* for the creation of the new schema, the retrieval of metadata and the completion of the new schema

with metadata values. That means, that the plug-in can be executed in any machine that a DSpace installation exists without the need of any other prerequisites to be installed.

Installation and Execution

The plug-in is provided through a *JAR* file than apart from the executable class also contains the source code. This *JAR* file must be installed in the DSpace installation folder. More precisely, the *JAR* file must be copied in the library folder of *DSpace*, that is: ***dpace-installation-dir/lib***.

Next, the user can execute the code of the *JAR* file with the following steps

- 1) Move , using the console in the folder: *dspace-installation-dir/bin* με:

```
cd dspace-installation-dir/bin
```

- 2) Run the command:

```
./dsrun gr.ekt.repositories.utils.SetEuropeanaMetadata user
```

where ***user*** is a valid DSpace administrator user.

If *user* is not valid and thus cannot be authenticated in DSpace, the plug-in fails and exits. Otherwise, appropriate messages appear in the screen, depending on the DSpace installation.

```
root@devtom:/usr/local/DSpace_BUILDS/pandektis_1.5_postgres_dev/bin
[root@devtom bin]# ./dsrun gr.ekt.repositories.utils.SetEuropeanaMetadata kstamatis@ekt.gr
Creating Context... for user
Creating metadata schema for Europeana
Schema seems to exist... returning!!
Filling items with Europeana Metadata
Iterating through items!
```

(In the above case, the *Europeana* schema was found in the repository, thus, there is no need for creation)

That is, if the *Europeana* schema does not exist in the repository, a new one is created and is filled with all the necessary metadata fields. On the other hand, if the schema is found on the repository, the plug-in just precedes with the completion of the metadata values.

ATTENTION: If the *Europeana* schema already exists in the repository and some of the metadata values are completed, this *plug-in* **REMOVES** all the metadata values of the *Europeana* schema before proceeding to the completion of them from the beginning.

The metadata fields that this plug in fills are:

Europeana.provider: This field is filled automatically with the value provided in the configuration file. The name of the parameter that is used for this reason is *Europeana.provider*. if this parameter

does not exist in the configuration file, then the plug-in inserts blank as the value of *europiana.provider*.

europiana.type: This field is filled automatically with the value provided from the parameter *europiana.type.default* from the configuration file or with the specified values in the parameters of type *europiana.type.****. (see the configuration parameters below). In case that parameter *europiana.type.default* has no value, then the default value *IMAGE* is used.

europiana.isShownAt: This field is filled automatically with the value derived from the concatenation of the *DSpace* url (*dspace.url* parameter from *DSpace* configuration file) and the item's handler.

europiana.isShownBy: This field is filled automatically with the value of the parameter *europiana.isshownby* in the configuration file.

Configuration

The plug-in configuration takes place in the *DSpace* configuration file located in:

```
dspace-installation-dir/config/dspace.cfg
```

The application can operate without any configuration. In this case, the default values are used in the completion of metadata.

At the end of the configuration file the user can add any of the following parameters:

europiana.provider: the value of this parameter will be used for the metadata field *europiana.provider*. If no value is given for this parameter, the metadata field remains blank.

i.e.:

```
europiana.provider = National Documentation Center (EKT)
```

europiana.type.default: This parameter stores the default value for the type that will be used in the *europiana.type* metadata field, in case that no value is specified in the parameters of type: *europiana.type.****. Even if this parameter is not specified, the default value for this parameter is *IMAGE*.

europiana.type.*:** In this case, the ***** can be replaced with any of the following keywords *image*, *text*, *sound*, *video*. The value of this parameter must be the handler of a collection and declares the type of the items that the specified collection contains. In this manner, the type fills the field *europiana.type*. For the same type, more than one collection can be provided, separated by comma.

i.e.:

```
eupeana.type.text = 123456789/23162
```

```
eupeana.type.sound = 123456789/327, 123456789/2
```

The above configuration means that the items of the collection with handler 123456789/23162 will be given the type of *TEXT* while the items of the collections with handlers 123456789/327 and 123456789/2 will be given the type of *SOUND*. All other items from not specified collections will be given the type that is specified in parameter *eupeana.type.default*, if exists, or *IMAGE* otherwise.

eupeana.isshownby: In this parameter the user can specify the value from a *DC* metadata field that wants to be passed in the eupeana *eupeana.isShownBy* field. The value of this parameter must have the following pattern:

```
include_baseurl:prefix:dc:element.qualifier
```

where:

- *include_baseurl* can have the values *TRUE* and *FALSE*, depending on if we want to include the dspace base url in front of the value (*dspace.url* parameter from *configuration* file of *DSpace – dspace.cfg*)
- *prefix* is a prefix that may need to exist between the *base url* and the *DC metadata value*. If no prefix exists, this part of configuration is omitted.
- *element* and *qualifier* specify the DC element and qualifier of the DC metadata that will be used as a value..

i.e.:

```
eupeana.isshownby = FALSE::dc:identifier.uri
```

Given that the field *dc:identifier.uri* has the value “**item_image.jpg**” the aforementioned configuration means that the field *eupeana.isShownBy* will be filled with the value **item_image.jpg**.

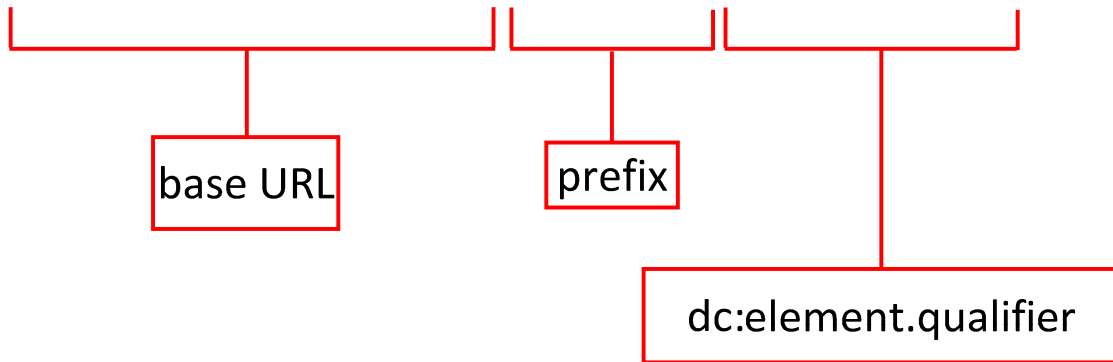
i.e.:

```
eupeana.isshownby = TRUE:bitstream/:dc:identifier.uri
```

Given that the field *dc:identifier.uri* has the value **item_image.jpg** and the base url of the dspace installation is **http://server.com/dspace/**, the aforementioned configurations means that the field *eupeana.isShownBy* will be filled with the value

http://server.com/dspace/bitstream/item_image.jpg as explained below:

http://server.com/dspace/bitstream/item_image.jpg



Attention: It may be the case that prefix is followed or not by a slash character / depending on the format of the value of the *dc:identifier.uri* metadata field.