

DOOR Manual – Import Object API

Adding an object in DOOR

Process

The process used to add an object to the repository is divided in 3 phases:

1. Partially add the object (you get the new object's id)
2. Upload attachment and rename it with the object id (for this reason you partially add the object before upload)
3. Complete the object

DB functions

The file 'door/lib/db/dbcom.php' contains definitions of functions used to interact with the db. In this file there are two function used to add an object to the repository:

1. `addobject(object obj)`
2. `upload the attachment`
3. `complete_object(integer object_id, object obj)`

The first function performs the first phase of the process, the second one does the third phase.

Structure of the object 'obj'

Add (and also import) procedure use a particular object. It's not defined (using a class). It's just a variable with many fields that can be read/overwritten quickly. For example, to define a user with name, age and phone number it's quite simple, do as follows:

```
$obj;  
$obj->name = "Mauro";  
$obj->age = 24;  
$obj->phone_number = "0791111111";
```

Concerning DOOR, an object can be added to the repository if it's defined as follows:

```
$obj;  
$obj->title;  
$obj->description;  
$obj->keywords;  
$obj->language;  
$obj->type_of_resource;  
$obj->useBy;  
$obj->useIn;  
$obj->required_time;  
$obj->time_insert;  
$obj->uid;  
$obj->logi_user_id;  
$obj->catalog_tree_id;  
...  
$obj->id (later defined, $obj->id = addobject($obj));
```

As you can see, the object has an id, a title, a description, keywords, languages, types of resource, use by items, use in items, required time to be 'digested' (learned), , id of the user that published it, the id of the node that will contains it.

Importing an object in DOOR

DOOR uses the functions defined in the file “door/lib/digest_object.php” to import contains the following functions used to import Learning Objects to the repository.

Functions

Privates functions (this functions should not be called directly):

- seems_utf8(string str)
this function checks if the character encoding is utf8
- iso88591_ensure(string str)
this function uses the previous function and decode a string from utf8 to iso88591
- recursive_remove_directory(string directory, bool empty)
this function removes a directory and its content
- clean(integer id, boolean delete_directory)
this function is used when an error occurs to remove unnecessary data.
- str_replace_count(string search, string replace, string subject, integer times)
this function replaces 'times' times 'search' with 'replace' in 'subject'.
- mkdir_p(string directory)
this recursive function creates a path, it's equivalent to the unix command “mkdir -p path”

Public functions (functions that must be called to import a LO):

- Functions defined to import all assets from Moodle (only used in this case):
 - prepareFile(integer id, \$file)
 - get_imslist(string path_to_meta_ims)
- Functions used to upload a LO:
 - uploadFile(integer id)
- Functions used in both cases:
 - digest_object(object obj)

Procedure to develop a plugin for another platform

IF you want to develop a plugin for another platform different from Moodle just do as follows:

- Create a new file 'import_object_from_YOURPLATFORM.php' under 'door/admin/'.
- Look at the files 'door/admin/import_object*', you'll find that they are divided in three parts:
 - copy the IMS file from YOURPLATFORM server to DOOR server (under the directory door/uploads/tmp). You need a mechanism in YOURPLATFORM that let you pick up the IMS file.
 - define a user interface to customize the import process (e.g. form that let you decide where to put the object)
 - elaborate the object (after form submission):
 - define an 'empty' object as explained before
 - call the necessary methods (addobject (\$obj->id = addobject(\$obj)), prepareFile, digest_object, complete_object)
 - show the page you want (for example the page that shows the details of the imported object)