Reptor Manual

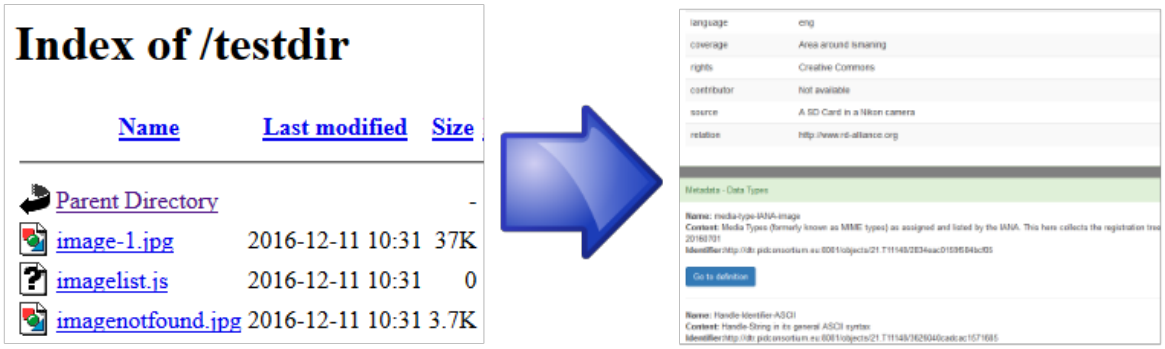
Thomas Zastrow, 2017

Table of Content

by Thomas Zastrow, 2017

1. What it is

Reptor is a PHP application which turns a webserver into a data repository. It demonstrates the functionality of a modern *data repository* along the recommendations of the Research Data Alliance (RDA)[1]. By adding *metadata* and *Persistent Identifiers* to a folder with *Object Data*, Reptor turns simple folders on a webserver into a *Digital Object*.



After beeing installed, Reptor offers:

- Untyped metadata handling by supporting several types of serialised metadata (plain text, key-value pairs [Dublin Core etc.], XML formats, etc.)
- PID handling, independent of the used PID system (DOI, Handle, URN etc.). The *shortref.org* service is preconfigured and can be used to assign Handle based PIDs to Reptor's Digital Objects
- Integration of Data Type Registries (DTR)
- Support for virtual collections (experimental)
- Access to the object data (files and folders) stored on the filesystem and taking care of permission management
- OAI-PMH
- ResourceSync (partially)
- Social media integration (Facebook, Twitter)
- Admin interface

All data is stored in one or more filesystems the webserver has access to. No database is necessary. Replication and backup is easy doable by just copying over the root folder of the Reptor data.

[1]https://www.rd-alliance.org

2. How it works

Reptor consists of a bunch of PHP, JavaScript and CSS files. The main work for the users view is done by the file **index.php** which takes some parameters and then displayes recursively the content of the folder called **data** as Digital Objects. Also external filesystems which are mounted into **data** are exposed. Metadata, PIDs etc. will be rendered automatically as well as dissemination technologies like OAI-PMH and social media buttons will be shown to the user. The admin interface can be used to create metadata and PIDs as well as uploading new data to the webserver.

3. Installation

Simply clone the GitHub repository into an empty folder on a PHP enabled webserver:

```
git clone https://github.com/TomZastrow/reptor.git
```

Beside the basic Reptor files, this will also create some basic example entries in the “data” folder: this folder is holding the whole data, any other (external) filesystem can be mounted on any mountpoint below “data”.

4. Using Docker

The Reptor software needs nothing else than a PHP enabled webserver to run. On DockerHub, you will find a bunch of images which could be used. Here is one way:

- Clone the Reptor software as mentioned in the chapter before from GitHub to your local machine
- Download the “eboraas/apache-php” Docker image:

  ```
  docker pull eboraas/apache-php
  ```
- Run the image and mount your local clone of the GitHub repository:

  ```
  docker run -p 9999:80 -v /path/to/git-clone/:/var/www/html -d eboraas/apache-php
  ```

Here, “9999” is the port on your host machine where the Reptor software is being made available. and “/path/to/git-clone” points to the directory where you made a clone of the Reptor GitHub repository. Now point your browser to:

http://localhost:9999

5. Cloud Storage Support

Reptors data just has to be stored somewhere below its „data“ directory. Here, any other (remote)

filesystem can be locally mounted. Reptor will then handle these filesystems just as local (sub-) directories. Make sure that the webserver has the file permissions to access the mounted filesystems.
Via this mechanism, a user can add / edit data easily on a remote filesystem while any change is instantaneously visible in Reptor.

6. Permissions

Permissions can be handled via the webserver. In the case of Apache, .htaccess can be used: just put a correct configured file in a subdiretcory of “data” and the Apache webserver will take care of the correct handling of its content.
Following RDAs recommendation: the metadata and the PID in this directory will be displayed always! Only accessing the content itself follows under the rules defined in .htaccess.

7. Configuration

Configuration is done in the file “config.php”. Most of these settings are coming along with a senseful default setting but can be adjusted accordingly to individual needs:

$dataTypeRegistry = "http://dtr.pidconsortium.eu:8081/objects/";

The URL to a data type registry: per default, GWDGs DTR is used

$namePIDFile = "pid.txt";

The name of the file which contains the PID of the current directory

$nameMetadataText = "metadata.txt";

The name of the file which contents is imterpreted and displayed as metadata text

$namesDataTypes = "metadata.types";

The name of the file which contains the data types, related to the data type registry mentioned above, to describe the content of the current directory

$nameDCFile = "metadata.dc";

This file contains Dublin Core metadata in Key-Value format

$template = "default";

The CSS- and HTML template which is used to render the pages. Can be found in subfolder “templates”.

$showEvaluation = true;

Defines if an evaluation against the RDA specifications is shown (true) or not (false) at the

bottom of the page

8. Functionality

9. Persistent Identifiers (PIDs)

Reptor is currently not able to manage PIDs on its own. You need an external PID provider, which can be a Handle based service (for example DOI) or any other PID system like ARK[2]. The PID should point to:

```
http://yourserver.com/path/to/reptor/subfolder/subfolder
```

where „subfolder/subfolder“ is the relative path to the folder you want to assign a PID to. Please note that the folder „data“ as the starting folder of the data repository is not part of the path. Also „index.php“ is not part because we assume that your server is configured in such a way that „index.php“ is delivered by default if no other filename is mentioned.

10.Metadata

Metadata has to be stored in files in any (sub-) directory of the „data“ dir. Depending on the filename, the metadata will be handled differently:

- **metadata.txt:** Its content will be displayed as it is in the „Metadata“ box of that directory. Any HTML, XML or whatever content is possible.
- **metadata.dc:** Key-value pairs as described in Dublin Core. Will be included in OAI-PMH output!
- **metadata.types:** semantic explanations of the content of the current directory. Makes use of the „Data Type Registry“ at GWDG

11.Virtual Collections

Reptor contains rudimentary support for creating and managing virtual collections. A virtual collection is a list of links and / or PIDs pointing to external resources. These links are shown as they are, no automatic resolving is done.
The file which contains the virtual collection is defined in the variable “nameCollectionItems”, per

[2]A good starting point is the free-of-cost service Shortref (www.shortref.org)

default this is “collections.txt”.

12.OAI-PMH

The “Open Archive Initiative - Protocol for Metadata Harvesting” (OAI-PMH) is implemented in Reptor and can be used to offer the metadata for other repositories, cataloge systems etc. for harvesting. The Reptor implementation supports the following OAI-PMH verbs:

- Identify
- ListMetadataFormats
- ListSets
- ListIdentifiers
- ListRecords

The OAI-PMH implementation is complete, it validates against the OAI-PMH validator at oaipmh.com[3]. Currently, it only offers “Dublin Core” metadata for harvesting.

13.ResourceSync

ResourceSync[4] is a specification for synchronization of resources between servers. It implementation in Reptor is rudimentary.

14.

[3] http://validator.oaipmh.com/

[4]http://www.openarchives.org/rs/1.0.9/resourcesync