

# Lesson: Microservices 101

## Part 2/2

**April 26, 2016**

( My name is Diego Pino Navarro  
I work at metro.org )

# What did we learn in Part 1/2

What are Microservices and  
Why we use them

Modern Php: namespaces and  
beyond

Basics of Composer

PSR-4 style autoloading

Quick intro to Silex

— — —

# What will we cover in Part 2/2

Using current Islandora  
CLAW microservices

Creating a new  
Microservice: Basic Image

---

# Using microservices in CLAW

# What we have right now

— — —

<https://github.com/Islandora-CLAW/CLAW/tree/sprint-002>



Inside /services

## ResourceService

Wrapper around Fedora4 RESTful API (CRUD)

Uses UUID to identify base Resources!

## TransactionService

Manages Batch Operations, a.k.a as TX

## CollectionService

CRU for PCDM Collections. Creates LDP containers, ore:Proxy, etc

See something missing? We are working!

**Using the all the services**

# Basic dev/test deployment

— — —

```
$ git clone -b sprint-002 https://github.com/Islandora-CLAW/CLAW.git
islandora-claw-micro
$ cd islandora-claw-micro/install
$ vagrant up (get a coffee)
```

This will start everything for you at port 8282!

# Let's test this

— — —

```
$ curl -i -XPOST -H "Slug:FirstCollection" http://localhost:8282/islandora/collection
```

```
HTTP/1.1 201 Created
Date: Tue, 26 Apr 2016 14:33:39 GMT
Server: Apache/2.4.20 (Ubuntu)
ETag: "6bd86a2f55c9f2d169fc13238055162e4412c796"
Cache-Control: private, must-revalidate
Last-Modified: Tue, 26 Apr 2016 14:33:47 GMT
Location: http://localhost:8282/islandora/resource/4d63ebb4-3f95-41f6-b087-bc684f1f9efd
Content-Length: 77
Connection: close
Link: <http://localhost:8282/islandora/resource/4d63ebb4-3f95-41f6-b087-bc684f1f9efd/members>; rel="hub"
Content-Type: text/plain; charset=UTF-8
```

## Magic/This will:

- Create a `pcdm:collection` object at fedora root
- add an `LDP indirect Container` to it named “members”
- Assign an UUID V4 to the `pcdm:collection`



# Let's use another service

```
$ curl -i -XGET http://localhost:8282/islandora/resource/4d63ebb4-3f95-41f6-b087-bc684f1f9efd
```

```
@prefix premis: <http://www.loc.gov/premis/rdf/v1#> .
```

```
...
```

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
```

```
<http://localhost:8080/fcrepo/rest/FirstCollection> a ldp:RDFSource , ldp:Container , <http://www.jcp.org/jcr/nt/1.0folder> , <http://www.jcp.org/jcr/nt/1.0hierarchyNode> , <http://www.jcp.org/jcr/nt/1.0base> , <http://www.jcp.org/jcr/mix/1.0created> , pcdm:Collection , fedora:Container , fedora:Resource , <http://www.jcp.org/jcr/mix/1.0lastModified> , <http://www.jcp.org/jcr/mix/1.0referenceable> ;
```

```
    fedora:lastModifiedBy "bypassAdmin"^^<http://www.w3.org/2001/XMLSchema#string> ;
```

```
    fedora:createdBy "bypassAdmin"^^<http://www.w3.org/2001/XMLSchema#string> ;
```

```
    isl:hasURN <urn:uuid:4d63ebb4-3f95-41f6-b087-bc684f1f9efd> ;
```

```
    fedora:primaryType "nt:folder"^^<http://www.w3.org/2001/XMLSchema#string> ;
```

```
    fedora:created "2016-04-26T14:33:42.072Z"^^<http://www.w3.org/2001/XMLSchema#dateTime> ;
```

```
    fedora:mixinTypes "pcdm:Collection"^^<http://www.w3.org/2001/XMLSchema#string> , "fedora:Container"^^<http://www.w3.org/2001/XMLSchema#string>
```

```
 , "fedora:Resource"^^<http://www.w3.org/2001/XMLSchema#string> ;
```

```
    fedora:lastModified "2016-04-26T14:33:47.117Z"^^<http://www.w3.org/2001/XMLSchema#dateTime> ;
```

```
    nfo:uuid "4d63ebb4-3f95-41f6-b087-bc684f1f9efd"^^<http://www.w3.org/2001/XMLSchema#string> ;
```

```
    fedora:writable "true"^^<http://www.w3.org/2001/XMLSchema#boolean> ;
```

```
    fedora:hasParent <http://localhost:8080/fcrepo/rest/> ;
```

```
    ldp:contains <http://localhost:8080/fcrepo/rest/FirstCollection/members> .
```

```
<http://localhost:8080/fcrepo/rest/FirstCollection/fcr:export?format=jcr/xml> dc:format <http://fedora.info/definitions/v4/repository#jcr/xml> .
```

```
<http://localhost:8080/fcrepo/rest/FirstCollection> fedora:exportsAs <http://localhost:8080/fcrepo/rest/FirstCollection/fcr:export?format=jcr/xml> .
```

# Like a Matryoshka doll

— — —

`CollectionService` uses internally

- `ResourceService`
- `TransactionService`

Also mounts the endpoints for those.

One Service exposes all other functionality via RESTful API!

# Let's build a new Service

With your help!

<https://raw.githubusercontent.com/wiki/duraspace/pcdm/yEd/islandora-basic-image/Islandora-PCDM-Basic-Image.inn>



# Skeleton Image Service Silex app [\(src/index.php\)](#)



```
<?php
namespace Islandora\ImageService;
require_once __DIR__.'../vendor/autoload.php';
use Silex\Application;
use Islandora\Chullo\Uuid\UuidGenerator;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpKernel\HttpKernelInterface;
use Psr\Http\Message\ResponseInterface;
use Silex\Provider\TwigServiceProvider;
use Islandora\ResourceService\Provider\ResourceServiceProvider;
use Islandora\CollectionService\Provider\CollectionServiceProvider;
use Islandora\TransactionService\Provider\TransactionServiceProvider;

date_default_timezone_set('UTC');
$app = new Application();
$app['debug'] = true;
$app->register(new \Silex\Provider\ServiceControllerServiceProvider());
$app->register(new \Silex\Provider\TwigServiceProvider(), array(
    'twig.path' => __DIR__.'../templates',
));
```

# Mount our depending MicroServices



```
$islandoraResourceServiceProvider = new ResourceServiceProvider;

//Registers Resource Service and defines current app's path for config context
$app->register($islandoraResourceServiceProvider, array(
    'islandora.BasePath' => __DIR__,
));
$app->mount("/islandora", $islandoraResourceServiceProvider);

$islandoraTransactionService = new TransactionServiceProvider;

$app->register($islandoraTransactionService);
$app->mount("/islandora", $islandoraTransactionService);
```

# Now define a new one for Image

— — —



```
$islandoraImageService = new ImageServiceProvider;  
  
$app->register($islandoraImageService, array(  
    'UuidGenerator' => new UuidGenerator(),  
));  
$app->mount("/islandora", $islandoraImageService);
```

# Manage default responses and errors (copy/pasta)



```
/**
 * Convert returned Guzzle responses to Symfony responses.
 */
$app->view(function (ResponseInterface $psr7) {
    return new Response($psr7->getBody(), $psr7->getStatusCode(), $psr7->getHeaders());
});

$app->after(function (Request $request, Response $response) use ($app) {
    $response->headers->set('X-Powered-By', 'Islandora Image REST API v' . $app['config']['islandora']['apiVersion'], TRUE); //Nice
});

//Common error Handling
$app->error(function (\EasyRdf_Exception $e, $code) use ($app) {
    if ($app['debug']) {
        return;
    }
    return new response(sprintf('RDF Library exception: %s', $e->getMessage()), $code), $code);
});
$app->error(function (\Symfony\Component\HttpKernel\Exception\HttpException $e, $code) use ($app) {
    if ($app['debug']) {
        return;
    }
    return new response(sprintf('Islandora Image Service exception: %s / HTTP %d response', $e->getMessage(), $code), $code);
});
$app->error(function (\Symfony\Component\HttpKernel\Exception\NotFoundHttpException $e, $code) use ($app) {
    if ($app['debug']) {
        return;
    }
    return new response(sprintf('Islandora Image Service exception: %s / HTTP %d response', $e->getMessage(), $code), $code);
});
$app->error(function (\Exception $e, $code) use ($app) {
    if ($app['debug']) {
        return;
    }
    return new response(sprintf('Islandora Image Service unatched exception: %s %d response', $e->getMessage(), $code), $code);
});
```



# And finally

— — —

```
$app->run();
```



# Now into the Service itself [\(src/Provider/ImageServiceProvider.php\)](#)

```
<?php

namespace Islandora\ImageService\Provider;

use Silex\Application;
use Silex\ServiceProviderInterface;
use Silex\ControllerProviderInterface;
use Islandora\Chullo\FedoraApi;
use Islandora\Chullo\TriplestoreClient;
use Islandora\Chullo\Uuid\UuidGenerator;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpKernel\HttpKernelInterface;
use Symfony\Component\Yaml\Yaml;
use Islandora\CollectionService\Controller\CollectionController;

class ImageServiceProvider implements ServiceProviderInterface, ControllerProviderInterface {
    ...WE WILL HAVE TO DEVELOP THIS PART!
}
```

**Let's switch to real code**

# Only a skeleton

Anyone willing to commit to Islandora CLAW?

# And derivatives?

---

