Share your information

Extending eZ Publish's REST API *Developer Preview #2*

By Łukasz Serwatka
http://serwatka.net
and
Ole Marius Smestad

Index

1 Goal description

At the end of this tutorial, you will be able to embed your custom RESTful interface into eZ Publish, to be further consumed by any 3rd party-service :

- mobile application
- external business application
- web service

This lets you expose any service reading content (heading for a more complete CRUD support eventually), as well as other, fully custom features, with direct access to about any eZ Publish feature, from PHP.

This tutorial is based on the second developer preview of eZ Publish's REST framework, still being evolved to reach its stable state around the Matterhorn (4.5) release. There may be a few rough edges. Authentication (OAuth) is supported, but was disabled in this 2nd developer preview.

2 Pre-requisites and target population

This tutorial is for developers who are keen on making their eZ Publish a REST engine for their specific needs. This 2nd developer preview follows a first one, with which we recommend you to quickly get acquainted. Developer preview means that what you will learn in a few moments is necessarily going to evolve, change, maybe not only slightly. This is not a crystallised piece of knowledge, rather a intermediate (yet pretty advanced) deliverable in an iterative process. Let us all be agile !

Important preliminary note:

We are highly recommending to setup REST layer in Virtual Host environment for this release. We are working on adding support for other environments as well. Stay tuned!

Read more on how to set-up your eZ Publish in Virtual Host mode : http://doc.ez.no/eZ-Publish/Technical-manual/4.4/Installation/Virtual-host-setup

3 Step 1 : Installation

Here are the extensions/tarballs you will build off of :

- ezp_rest_preview_2.tgz (to be downloaded from the web version of this tutorial)
- ezxrestapidemo.zip (to be downloaded from the web version of this tutorial)

Unpack the first one somewhere, and follow the instructions in the INSTALL file, found at the root of what you just unpacked.

The second tarball above is actually what you should end-up with at the end of this tutorial. We recommend you build it yourself, following the steps below, much better to understand what you are doing. If you are in a super-rush though, 25 minutes away from the Christmas diner (random example), you may want to download it :)

4 Step 2 : Understanding the REST URI pattern

The resource URIs provided by default by the eZ Publish REST extension support the following convention:

```
/<prefix>/ + <provider>/<version> + /<function>/<params>
```

For the built-in REST API, the default provider name is "ezp". The version token is build as "v + integer" for example v1 for REST API version one. The global prefix (first URI part) can be defined in the REST configuration as settings/rest.ini.[System].ApiPrefix.

The RegExp based URI filtering is handled by a default "ezpRestDefaultRegexpPrefixFilter" class implementation where <provider> and <version> data are used for routs filtering. Developers can implement custom filters by implementing the "ezpRestPrefixFilterInterface" in first place and then updating the "rest.ini.[System].PrefixFilterClass" accordingly. Getting implementation ideas out of the "ezpRestDefaultRegexpPrefixFilter" can be a good start.

5 Step 3 : Understanding REST API versioning

The resource URIs contain a version token which is build as "v + integer", for example v1 for REST API version one, v2 for version two and so on. Based on the version information provided in the resource URI the pre-routing filter extracts version information and matches against defined versioned routes. The "ezpRestVersionedRoute" is a route wrapper around existing instance of "ezcMvcRoute" object providing multiple versions of it. Find an example of a versioned routes definition below:

URI: /api/provider/v1/foo

```
new ezpRestVersionedRoute( new ezcMvcRailsRoute( '/foo', 'ezxRestController', 'foo' ), 1 )
```

URI: /api/provider/v2/foo

```
new ezpRestVersionedRoute( new ezcMvcRailsRoute( '/foo', 'ezxRestController', 'fooBar' ), 2 )
```

For the first URI ezxRestController::doFoo() method will be executed, which is version one. For the second URI which is a version two, ezxRestController::doFooBar()will be called. Both methods can share implementation logic but differ with the output result for instance.

6 Step 4 : Extending REST

As of now it is possible to extend the eZ Publish REST interface with custom functionality. One REST extension can support many different data providers as well as different versions. The following steps presents how to create a new eZ Publish REST extension.

1. Create an empty folder called "ezxrestapidemo" under the " <eZ Publish>/extension" location.

2. Inside "ezxrestapidemo" create a setting with the "rest.ini.append.php" file with following content:

```
<?php /*

[ApiProvider]

ProviderClass[ezx]=ezxRestApiProvider

*/ ?>
```

ProviderClass is an array where the index is a provider token used in the resource URI. For example for URI: "/api/ezx/v1/foo" the registered provider with token "ezx" will be called. You can have many providers registered within one extension.

3. Next create a folder called "classes" under the "<eZ Publish>/extension/ezxrestapidemo/" location with the "rest_provider.php file". Edit the newly created PHP file and add the following code:

```
<?php

class ezxRestApiProvider implements ezpRestProviderInterface
{
    /**
     * Returns registered versioned routes for provider
     *
     * @return array
     */
    public function getRoutes()
    {
        return array( new ezpRestVersionedRoute( new ezcMvcRailsRoute( '/foo',
                                                     'ezxRestController', 'foo' ), 1 ),

                      new ezpRestVersionedRoute( new ezcMvcRailsRoute( '/foo',

                                                      'ezxRestController', 'fooBar' ), 2 ) );
    }

    /**
     * Returns associated with provider view controller
     *
     * @return ezpRestViewController
     */
    public function getViewController()
    {
        return new ezxRestApiViewController();
    }
}
?>
```

Every REST API provider has to implement the "ezpRestProviderInterface" where "getRoutes()" methods returns registered versioned routes for provider and "getViewController()" returns view controller object which has to implement the "ezpRestViewControllerInterface".

4. In the next step create the file “view_controller.php” under the “<eZ Publish>/extension/ezxrestapidemo/classes” folder with following code:

```
<?php

class ezxRestApiViewController implements ezpRestViewControllerInterface
{
    /**
     * Creates a view required by controller's result
     *
     * @param ezcMvcRoutingInformation $routeInfo

    * @param ezcMvcRequest $request
    * @param ezcMvcResult $result
    * @return ezcMvcView
    */
    public function loadView( ezcMvcRoutingInformation $routeInfo, ezcMvcRequest $request,
ezcMvcResult $result )
    {
        return new ezpRestJsonView( $request, $result );
    }
}
?>
```

Every view controller has to implement the “ezpRestViewControllerInterface” where the “loadView()” method returns an instance of the “ezcMvcView” object. In this example we re-use the “ezpRestJsonView” which is a part of the built-in API, if your provider need a custom view, you can return it here.

5. Registered for REST provider URIs are associated with controllers. This example calls the methods foo() and fooBar() on “ezxRestController”. Create a file called “rest_controller.php” under “<eZ Publish>/extension/ezxrestapidemo/classes” folder and put the following code inside:

```
<?php

class ezxRestController extends ezcMvcController
{
    public function doFoo()
    {
        $res = new ezcMvcResult();
        $res->variables['message'] = "This is FOO!";
        return $res;
    }

    public function doFooBar()
    {
        $res = new ezcMvcResult();
        $res->variables['message'] = "This is FOOBAR!";
        return $res;
    }
}

?>
```

Notice the controller methods naming convention. All methods have to be prefixed with “do” word. This is a “ezcMvcController” requirement.

6. Enable “ezxrestapidemo” by adding the following to “ActiveExtensions” list in the “settings/override/site.ini.append.php”

```
[ExtensionSettings]
[…]
ActiveExtensions[]=oauth
ActiveExtensions[]=rest
ActiveExtensions[]=ezprestapiprovider
ActiveExtensions[]=ezxrestapidemo
```

7. Update the autoload array with a following command from the eZ Publish root folder :

```
       php bin/php/ezpgenerateautoloads.php -e -p
```

8. If you have followed all steps carefully then you should be ready to test your new REST API provider by calling www.example.com/api/ezx/v1/foo A simple JSON response should look like :

```
{"message":"This is FOO!"}
```

For the second URI www.example.com/api/ezx/v2/foo output should be following:

```
{"message":"This is FOOBAR!"}
```

7 Conclusion

You should now have the ezxrestapidemo extension ready, exposing your custom REST api to any REST-enabled channel. This tutorial showed the principle of extending the eZ Publish REST framework. Given the “preview” nature of this framework, some changes might occur in the future on the way REST extensions are done, stay tuned for all details. However, the foundations are now laid, let us build off of them together. You can now already start using it for leveraging the multi-channel approach. A few links in the Resources part to get you started building mobile apps.

Please share your feedback, impressions, ideas as comments under this post, that is how we move on together!

8 Resources

- First developer preview
- REST on Wikipedia
- iPhone app tutorials here and there
- Android app tutorial : here and there

9 About the authors : Lukasz & Ole Marius

Łukasz Serwatka is a Software Engineer at eZ Systems, focusing on dedicated eZ Publish solutions such as eZ Flow and the Website Interface.
http://serwatka.net

Ole Marius joined eZ Systems in 2005. He holds a master of technology from the the Norwegian University of Science and Technology. He is currently serving as domain leader for eZ Publish.

10 License choice