AnyBlok / Pyramid Documentation

Release 0.2.0

Jean-Sébastien Suzanne

June 27, 2016

Contents

1 Front Matter **3**
- 1.1 Project Homepage . . . 3
- 1.2 Project Status . . . 3
- 1.3 Installation . . . 3
- 1.4 Unit Test . . . 3
- 1.5 Dependencies . . . 4
- 1.6 Contributing (hackers needed!) . . . 4
- 1.7 Author . . . 4
- 1.8 Contributors . . . 4
- 1.9 Bugs . . . 4

2 MEMENTO **7**
- 2.1 Pyramid `route` and `view` which does not depend of the bloks . . . 7
- 2.2 Pyramid controller which depend of the installation of the bloks . . . 8

3 AnyBlok / Pyramid framework **15**
- 3.1 AnyBlok/ Pyramid controllers . . . 15
- 3.2 anyblok_pyramid.handler . . . 24
- 3.3 anyblok_pyramid.config . . . 25
- 3.4 anyblok_pyramid.scripts module . . . 25

4 Bloks **27**
- 4.1 `pyramid` blok . . . 27

5 Helper for unittest **29**
- 5.1 PyramidTestCase . . . 29
- 5.2 PyramidDBTestCase . . . 29
- 5.3 PyramidBlokTestCase . . . 29

6 ROADMAP **31**
- 6.1 To implement . . . 31
- 6.2 Functionnality which need a sprint . . . 31

7 CHANGELOG **33**
- 7.1 Future . . . 33
- 7.2 0.1.0 . . . 33

8 Mozilla Public License Version 2.0 **35**
- 8.1 1. Definitions . . . 35

8.2 2. License Grants and Conditions 37
8.3 3. Responsibilities 38
8.4 4. Inability to Comply Due to Statute or Regulation 39
8.5 5. Termination 39
8.6 6. Disclaimer of Warranty 39
8.7 7. Limitation of Liability 40
8.8 8. Litigation 40
8.9 9. Miscellaneous 40
8.10 10. Versions of the License 40
8.11 Exhibit A - Source Code Form License Notice 41
8.12 Exhibit B - "Incompatible With Secondary Licenses" Notice 41

9 Indices and tables **43**

Python Module Index **45**

Contents:

Contents

- *Front Matter*
  - *Project Homepage*
  - *Project Status*
  - *Installation*
  - *Unit Test*
  - *Dependencies*
  - *Contributing (hackers needed!)*
  - *Author*
  - *Contributors*
  - *Bugs*

CHAPTER 1

Front Matter

Information about the AnyBlok / Pyramid project.

1.1 Project Homepage

AnyBlok is hosted on Bitbucket - the main project page is at https://bitbucket.org/jssuzanne/anyblok_pyramid. Source code is tracked here using Mercurial.

Releases and project status are available on Pypi at http://pypi.python.org/pypi/anyblok_pyramid.

The most recent published version of this documentation should be at http://doc.pyramid.anyblok.org.

1.2 Project Status

AnyBlok with Pyramid is currently in beta status and is expected to be fairly stable. Users should take care to report bugs and missing features on an as-needed basis. It should be expected that the development version may be required for proper implementation of recently repaired issues in between releases; the latest master is always available at http://bitbucket.org/jssuzanne/anyblok_pyramid/get/default.tar.gz.

1.3 Installation

Install released versions of AnyBlok from the Python package index with pip or a similar tool:

```
pip install anyblok_pyramid
```

Installation via source distribution is via the `setup.py` script:

```
python setup.py install
```

Installation will add the `anyblok` commands to the environment.

1.4 Unit Test

Run the test with `nose`:

```
pip install nose
nosetests anyblok_pyramid/tests
```

1.5 Dependencies

AnyBlok works with **Python 3.2** and later. The install process will ensure that AnyBlok, Pyramid are installed, in addition to other dependencies. The latest version of them is strongly recommended.

1.6 Contributing (hackers needed!)

Anyblok / Pyramid is at a very early stage, feel free to fork, talk with core dev, and spread the word!

1.7 Author

Jean-Sébastien Suzanne

1.8 Contributors

Anybox team:

- Georges Racinet
- Christophe Combelles
- Sandrine Chaufournais
- Jean-Sébastien Suzanne
- Florent Jouatte
- Simon André
- Clovis Nzouendjou
- Pierre Verkest
- Franck Bret

1.9 Bugs

Bugs and feature enhancements to AnyBlok should be reported on the Issue tracker.

Contents

- *MEMENTO*
  - *Pyramid `route` and `view` which does not depend of the bloks*
  - *Pyramid controller which depend of the installation of the bloks*
    - *HTTP controller*
    - *JSON-RPC controller*
    - *XML-RPC controller*
    - *Pyramid `Mixin`*
    - *Inheritance*
    - *Pyramid `Core`*
    - *Cache*
    - *Properties*

CHAPTER 2

MEMENTO

Anyblok / Pyramid mainly depends on:

- Python 3.2+
- AnyBlok
- Pyramid

If the scrip `anyblok_wsgi` is used to start the `WSGI` application, the you can not declare `route` and `view`. AnyBlok / Pyramid define two familly of controller:

- Controller which no depend of blok
- Controller which depend of the installation or not of bloks

2.1 Pyramid `route` and `view` which does not depend of the bloks

The goal is to declare in your application code source the `route` and the `view`:

```
from anyblok import Declarations
Pyramid = Declarations.Pyramid
```

Declare a `view`:

```
@Pyramid.add_view('route name')
def myview(request):
    ...
```

Note: The decorator `add_view` is just a wrapper of add_view

the args already filled by the wraper are:

- view: is the decorated function
- name: is the **route name**

Declare a `route`:

```
Pyramid.add_route('route name', '/my/path')
```

Note: The function `add_route` is just a wrapper of add_route

The args already filled by the wraper are:

- name: is the **route name**
- pattern: is the path

Warning: It 's important to use the add_route of Pyramid, because when the view are add in configuration, this view check is the **route name** exist in the routes.

2.2 Pyramid controller which depend of the installation of the bloks

Theses controllers must be declared in the bloks

The declaration of theses controllers is as the declaration of AnyBlok Model

They are three controllers which can be declared in the bloks:

- PyramidHTTP
- PyramidJsonRPC
- PyramidXmlRPC

The controller can be inherited by `Mixin`

- PyramidMixin

The controller inherit also `Core` and have some feature as:

- Cache
- Properties

2.2.1 HTTP controller

Get the `Type` of controller:

```
from anyblok import Declarations
PyramidHTTP = Declarations.PyramidHTTP
register = Declarations.register
```

Declare a `view`:

```
@register(PyramidHTTP)
class MyController:

    @PyramidHTTP.view()
    def myview(request):
        # route name == myview
        ...

    @PyramidHTTP.view('myroute')
    def myotherview(request):
        # route name == myroute
        ...
```

Note: The decorator `view` is just a wrapper of add_view

the args already filled by the wraper are:

- view: is the decorated function
- name: the default value is the name of the method or the first args

Declare a `route`:

```
PyramidHTTP.add_route('route name', '/my/path')
```

Note: The function `add_route` is just a wrapper of add_route

The args already filled by the wraper are:

- name: is the **route name**
- pattern: is the path

Warning: It 's important to use the add_route of PyramidHTTP, because when the view are add in configuration, this view check is the **route name** exist in the routes.

2.2.2 JSON-RPC controller

Get the `Type` of controller:

```
from anyblok import Declarations
PyramidJsonRPC = Declarations.PyramidJsonRPC
register = Declarations.register
```

Declare a `rpc method`:

```
@register(PyramidJsonRPC)
class MyController:

    @PyramidJsonRPC.rpc_method()
    def mymethod(request):
        # method name == mymethod
        ...

    @PyramidJsonRPC.rpc_method('myroute')
    def myothermethod(request):
        # method name == myroute
        ...
```

Note: The decorator `rpc_method` is just a wrapper of add_jsonrpc_method

the args already filled by the wraper are:

- view: is the decorated method
- **endpoint: the default value is the name of the method or the first** args

Declare a `route`:

```
PyramidJsonRPC.add_route(PyramidJsonRPC.MyController, '/my/path')
```

Note: The function `add_route` is just a wrapper of add_jsonrpc_endpoint

The args already filled by the wraper are:

- name: is the **route name**
- pattern: is the path

Warning: It 's important to use the add_route of PyramidJsonRPC, because when the view are add in configuration, this view check is the **rpc method** exist in the routes.

2.2.3 XML-RPC controller

Get the `Type` of controller:

```
from anyblok import Declarations
PyramidXmlRPC = Declarations.PyramidXmlRPC
register = Declarations.register
```

Declare a `rpc method`:

```
@register(PyramidXmlRPC)
class MyController:

    @PyramidXmlRPC.rpc_method()
    def mymethod(request):
        # method name == mymethod
        ...

    @PyramidXmlRPC.rpc_method('myroute')
    def myothermethod(request):
        # method name == myroute
        ...
```

Note: The decorator `rpc_method` is just a wrapper of add_xmlrpc_method

the args already filled by the wraper are:

- view: is the decorated method
- **endpoint: the default value is the name of the method or the first** args

Declare a `route`:

```
PyramidXmlRPC.add_route(PyramidXmlRPC.MyController, '/my/path')
```

Note: The function `add_route` is just a wrapper of add_xmlrpc_endpoint

The args already filled by the wraper are:

- name: is the **route name**
- pattern: is the path

Warning: It 's important to use the add_route of PyramidXmlRPC, because when the view are add in configuration, this view check is the **rpc method** exist in the routes.

2.2.4 Pyramid `Mixin`

Mixin is used to define behaviours on the controllers.

Declare a `Mixin`:

```python
from anyblok import Declarations
register = Declarations.register
PyramidMixin = Declarations.PyramidMixin


@register(PyramidMixin)
class MyMixin:
    ...
```

Inherit a `Mixin` by a controller:

```python
@register(PyramidHTTP)
class MyController(PyramidMixin.MyMixin):
    ...
```

Inherit a `Mixin` by another `Mixin`:

```python
@register(PyramidMixin)
class MyAnotherMixin(PyramidMixin.MyMixin):
    ...
```

2.2.5 Inheritance

The conbroller can inherit `PyramidMixin` and also Controller of the same `Type`:

```python
@register(PyramidHTTP)
class MyController(PyramidHTTP.OtherController):
    ...
```

2.2.6 Pyramid `Core`

The `Core` used by the controller are:

- ControllerBase: For all the controller
- ControllerHTTP
- ControllerRPC
- ControllerJsonRPC
- ControllerXmlRPC

Overload a `Core`:

```
@register(Core)
class ControllerBase:
    ...
```

2.2.7 Cache

Add a cache on a controller is as cache on a model.

Declare a cache on a controller:

```
@register(PyramidHTTP):
class MyController:

    @classmethod_method()
    def mycachedmethod(cls):
        ...
```

Declare a cache on a `Mixin`:

```
@registry(PyramidMixin)
class MyMixin:

    @classmethod_method()
    def mycachedmethod(cls):
        ...

@register(PyramidHTTP):
class MyController(PyramidMixin.MyMixin):
    ...
```

Declare a cache on a `Core`:

```
@registry(Core)
class PyramidBase:

    @classmethod_method()
    def mycachedmethod(cls):
        ...

@register(PyramidHTTP):
class MyController:
    ...
```

Warning: The instance of controller are not the same for each call. Then use `Declarations.cache` to cache in only one request else use `Declarations.classmethod_cache` to cache a method for all the request

2.2.8 Properties

the decorator `*Controller*.check_properties` allow to define an property to check before the `view` or `rpc_method` be called.

This property check if the *user* is authentificated:

```
@register(PyramidHTTP)
class MyController:
    def check_property_myproperty(self, value):
        """If the value property is not good the this method must raise"""

    @check_properties(myproperty=OneValue)
    @PyramidHTTP.view()
    def myview(self):
        ...
```

You can add your property but the property must be associated at a check method on the controller. This method can be in a `Mixin` or in a `Core`. This method can be overload.

Contents

- *AnyBlok / Pyramid framework*
  - *AnyBlok/ Pyramid controllers*
  - *anyblok_pyramid.handler*
  - *anyblok_pyramid.config*
  - *anyblok_pyramid.scripts module*

CHAPTER 3

AnyBlok / Pyramid framework

3.1 AnyBlok/ Pyramid controllers

class `anyblok_pyramid.controllers.`**`PyramidException`**

Exception for web type

class `anyblok_pyramid.controllers.`**`PyramidMixin`**

Bases: `anyblok.mixin.MixinType`

The PyramidMixin class are used to define a behaviours on models:

•Add new mixin class:

```
@Declarations.register(Declarations.PyramidMixin)
class MyMixinclass:
    pass
```

•Remove a mixin class:

```
Declarations.unregister(Declarations.PyramidMixin.MyMixinclass,
                        MyMixinclass)
```

class `anyblok_pyramid.controllers.`**`Pyramid`**

Bases: `object`

The Pyramid controller is a simple wrapper of the Pyramid controller

Pyramid can scan easily the `view` declarations to add them in the configuration. But the `route` have to add directlly in the configuration. This controller do all of them. The `route` and `view` are saved in the controller and the controller add them in the configuration at the start of the wsgi server

Warning: This case is only use by the script `anyblok_wsgi`, if you use an another script, you must include the includem `pyramid_config` or use the function `make_config` to get all the configuration

This `Type` is not an entry, no class are assembled in the registry. You must not add any class of this `Type`, the methods `register` and `unregister` raise an exception.

Add a view:

```
from anyblok import Declarations


@Declarations.Pyramid.add_view('route name')
```

```
def myview(request):
    ...
```

Note: The decorator `add_view` is just a wrapper of add_view

the args already filled by the wraper are:

- view: is the decorated function
- name: is the **route name**

Add a route:

```
from anyblok import Declarations


Declarations.Pyramid.add_route('route name', '/my/path')
```

Note: The function `add_route` is just a wrapper of add_route

The args already filled by the wraper are:

- name: is the **route name**
- pattern: is the path

Warning: It 's important to use the add_route of Pyramid, because when the view are add in configuration, this view check is the **route name** exist in the routes.

classmethod `add_route`(**args*, ***kwargs*)

Declare a route to add it in the configuration of `Pyramid`:

```
from anyblok import Declarations


Declarations.Pyramid.add_route('route name', '/my/path')
```

Note: The function `add_route` is just a wrapper of add_route

The args already filled by the wraper are:

- name: is the **route name**
- pattern: is the path

classmethod `add_view`(*endpoint*, ***kwargs*)

Declare a view to add it in the configuration of `Pyramid`:

```
from anyblok import Declarations


@Declarations.Pyramid.add_view('route name')
def myview(request):
    ...
```

Note: The decorator `add_view` is just a wrapper of add_view

the args already filled by the wraper are:

- view: is the decorated function
- name: is the **route name**

classmethod **`register`**(*parent*, *name*, *cls_*)

Forbidden method, this method always raise when calls

Parameters

- **`parent`** – Existing global registry
- **`name`** – Name of the new registry to add it
- **`cls`** – Class Interface to add in registry

Exception PyramidException

`routes` = []

Route properties to add in pyramid configuration

classmethod **`unregister`**(*child*, *cls_*)

Forbidden method, this method always raise when calls

Parameters

- **`entry`** – entry declaration of the model where the `cls_` must be removed
- **`cls`** – Class Interface to remove in registry

Exception PyramidException

`views` = []

View properties to add in pyramid configuration

class `anyblok_pyramid.controllers.`**`PyramidBase`**

Bases: `object`

Warning: This class is not a controller, but base of HTTP and RPC controller

Warning: This class is not the `Core.PyramidBase`.

The declarations of HTTP and RPC controller is not the same, but they are few difference.

classmethod **`assemble_callback`**(*registry*)

Assemble callback is called to assemble all the controllers from the installed bloks

Parameters **`registry`** – registry to update

classmethod **`authentificated`**()

Decorator which add the property `authentificated` with the value `True`

classmethod **`check_properties`**(***kwargs*)

decorator which add the properties to check

Parameters **`**kwargs`** – dict property: value to check

classmethod **`hook_insert_in_bases`**(*registry*, *bases*)

The difference between HTTP and RPC controller are the Core used by them. all of them must inherit of:

•Core.PyramidBase

•registry_base

Parameters

- **registry** – the current registry for the controller
- **bases** – bases list which define the controller

classmethod load_namespace(*registry*, *namespace*, *realregistryname=None*)

Return the bases and the properties of the namespace

Parameters

- **registry** – the current registry
- **namespace** – the namespace of the model
- **realregistryname** – the name of the model if the namespace is a mixin

Return type the list od the bases and the properties

Exception PyramidException

classmethod properties_from_decorators(*registryname*, *cls_*)

Properties is used to make some check before call the view. This method get the view which are need this verification

Parameters

- **registryname** – the registry name
- **cls** – a class of the registry name to take the properties

Return type dict to save in the registry

classmethod register(*parent*, *name*, *cls_*, ***kwargs*)

add new sub registry in the registry

Parameters

- **parent** – Existing global registry
- **name** – Name of the new registry to add it
- **cls** – Class Interface to add in registry

classmethod transform_base(*registry*, *namespace*, *base*, *properties*)

Detect specific declaration which must define by registry

Parameters

- **registry** – the current registry
- **namespace** – the namespace of the controller
- **base** – One of the base of the controller
- **properties** – the properties of the controller

Return type new base

classmethod unregister(*entry*, *cls_*)

Remove the Interface from the registry

Parameters

- **entry** – entry declaration of the model where the `cls_` must be removed

- **cls** – Class Interface to remove in registry

class `anyblok_pyramid.controllers.`**`PyramidHTTP`**

Bases: `anyblok_pyramid.controllers.PyramidBase`

The PyramidHTTP controller is a simple wrapper of the Pyramid controller

At the start of the pyramid server, all routes and all the views must be known. But the routes and views are declared on the bloks. Then the declaration of the routes and the views must be done also if the bloks are not installed. When the controller is called then the view must be validated by the controller to be called

Warning: This case is only use by the script `anyblok_wsgi`, if you use an another script, you must include the includem `pyramid_http_config` or use the function `make_config` to get all the configuration

Add a view:

```
from anyblok import Declarations


@Declarations.register(Declaration.PyramidHTTP)
class MyController:

    @Declaration.PyramidHTTP.view()
    def myview(request):
        # route name == myview
        ...

    @Declaration.PyramidHTTP.view('myroute')
    def myotherview(request):
        # route name == myroute
        ...
```

Note: The decorator `view` is just a wrapper of add_view

the args already filled by the wraper are:

- view: is the decorated function
- name: the default value is the name of the method or the first args

Add a route:

```
from anyblok import Declarations


Declarations.PyramidHTTP.add_route('route name', '/my/path')
```

Note: The function `add_route` is just a wrapper of add_route

The args already filled by the wraper are:

- name: is the **route name**
- pattern: is the path

Warning: It 's important to use the add_route of PyramidHTTP, because when the view are add in configuration, this view check is the **route name** exist in the routes.

classmethod add_route(**args, **kwargs*)
Declare a route to add it in the configuration of `Pyramid`:

```
from anyblok import Declarations


Declarations.PyramidHTTP.add_route('route name', '/my/path')
```

Note: The function `add_route` is just a wrapper of add_route

The args already filled by the wraper are:

- name: is the **route name**
- pattern: is the path

classmethod hook_insert_in_bases(*registry, bases*)
Define the Core class inherited by PyramidHTTP controllers

- Core.PyramidBaseHTTP
- super()

Parameters

- **registry** – the current registry for the controller
- **bases** – bases list which define the controller

classmethod hook_view_from_decorators(*registryname, cls_*)
Save the decorated method by view

Parameters

- **registryname** – registry name of the controller
- **cls_** – the cls of the registry name

Return type dict {'views': {route name: function} }

routes = []
Route properties to add in pyramid configuration

classmethod view(***kwargs*)
Declare a view to add it in the configuration of `Pyramid`:

```
from anyblok import Declarations


@Declarations.register(Declaration.PyramidHTTP)
class My controller:

    @Declaration.PyramidHTTP.view()
    def myview(request):
        # route name == myview
        ...

    @Declaration.PyramidHTTP.view('myroute')
```

```
            def myotherview(request):
                # route name == myroute
                ...
```

Note: The decorator `view` is just a wrapper of add_view

the args already filled by the wraper are:

- view: is the decorated function
- name: the default value is the name of the method or the first args

views = {}
View properties to add in pyramid configuration

class `anyblok_pyramid.controllers.`**`PyramidRPC`**
Bases: `anyblok_pyramid.controllers.PyramidBase`

classmethod **`add_route`**(**args*, ***kwargs*)
Declare a route to add it in the configuration of `Pyramid`

classmethod **`hook_insert_in_bases`**(*registry*, *bases*)
Define the Core class inherited by Pyramid RPC controllers

- Core.PyramidBaseRPC
- super()

Parameters

- **registry** – the current registry for the controller
- **bases** – bases list which define the controller

classmethod **`hook_view_from_decorators`**(*registryname*, *cls_*)
Save the decorated method by rpc method

Parameters

- **registryname** – registry name of the controller
- **cls_** – the cls of the registry name

Return type dict {'views': {route name: function} }

classmethod **`rpc_method`**(***kwargs*)
Declare a rpc method to add it in the configuration of `Pyramid RPC`

class `anyblok_pyramid.controllers.`**`PyramidJsonRPC`**
Bases: `anyblok_pyramid.controllers.PyramidRPC`

The PyramidJsonRPC controller is a simple wrapper of the Pyramid JSON-RPC controller

At the start of the pyramid server, all routes and all the rpc methods must be known. But the routes and rpc methods are declared on the bloks. Then the declaration of the routes and the rpc methods must be done also if the bloks are not installed. When the controller is called then the rpc method must be validated by the controller to be called

Warning: This case is only use by the script `anyblok_wsgi`, if you use an another script, you must include the includem `pyramid_jsonrpc_config` or use the function `make_config` to get all the configuration

Add a rpc method:

```
from anyblok import Declarations


@Declarations.register(Declaration.PyramidJsonRPC)
class MyController:

    @Declaration.PyramidJsonRPC.rpc_method()
    def mymethod(request):
        # method name == mymethod
        ...

    @Declaration.PyramidJsonRPC.rpc_method('myroute')
    def myothermethod(request):
        # method name == myroute
        ...
```

Note: The decorator `rpc_method` is just a wrapper of add_jsonrpc_method

the args already filled by the wraper are:

- view: is the decorated method
- **endpoint: the default value is the name of the method or the first** args

Add a route:

```
from anyblok import Declarations


Declarations.PyramidJsonRPC.add_route(
    Declarations.PyramidJsonRPC.MyController, '/my/path')
```

Note: The function `add_route` is just a wrapper of add_jsonrpc_endpoint

The args already filled by the wraper are:

- name: is the **route name**
- pattern: is the path

Warning: It 's important to use the add_route of PyramidJsonRPC, because when the view are add in configuration, this view check is the **rpc method** exist in the routes.

classmethod `hook_insert_in_bases` (*registry*, *bases*)

Define the Core class inherited by PyramidJsonRPC controllers

- Core.PyramidBaseJsonRPC
- super()

Parameters

- **`registry`** – the current registry for the controller
- **`bases`** – bases list which define the controller

methods = {}
RPC method properties to add in pyramid configuration

routes = []
Route properties to add in pyramid configuration

class `anyblok_pyramid.controllers.`**`PyramidXmlRPC`**
Bases: `anyblok_pyramid.controllers.PyramidRPC`

The PyramidXmlRPC controller is a simple wrapper of the Pyramid XML-RPC controller

At the start of the pyramid server, all routes and all the rpc methods must be known. But the routes and rpc methods are declared on the bloks. Then the declaration of the routes and the rpc methods must be done also if the bloks are not installed. When the controller is called then the rpc method must be validated by the controller to be called

Warning: This case is only use by the script `anyblok_wsgi`, if you use an another script, you must include the includem `pyramid_xmlrpc_config` or use the function `make_config` to get all the configuration

Add a rpc method:

```
from anyblok import Declarations


@Declarations.register(Declaration.PyramidXmlRPC)
class MyController:

    @Declaration.PyramidXmlRPC.rpc_method()
    def mymethod(request):
        # method name == mymethod
        ...

    @Declaration.PyramidXmlRPC.rpc_method('myroute')
    def myothermethod(request):
        # method name == myroute
        ...
```

Note: The decorator `rpc_method` is just a wrapper of add_xmlrpc_method

the args already filled by the wraper are:

- view: is the decorated method
- **endpoint: the default value is the name of the method or the first** args

Add a route:

```
from anyblok import Declarations


Declarations.PyramidXmlRPC.add_route(
    Declarations.PyramidXmlRPC.MyController, '/my/path')
```

Note: The function `add_route` is just a wrapper of add_xmlrpc_endpoint

The args already filled by the wraper are:

- name: is the **route name**

•pattern: is the path

Warning: It 's important to use the add_route of PyramidXmlRPC, because when the view are add in configuration, this view check is the **rpc method** exist in the routes.

classmethod hook_insert_in_bases (*registry*, *bases*)
Define the Core class inherited by PyramidXmlRPC controllers

•Core.PyramidBaseXmlRPC

•super()

Parameters

- **registry** – the current registry for the controller
- **bases** – bases list which define the controller

methods = {}
RPC method properties to add in pyramid configuration

routes = []
Route properties to add in pyramid configuration

3.2 anyblok_pyramid.handler

class anyblok_pyramid.handler.**Handler**
Base class for all the pyramid handler.

call_controller (**args*, ***kwargs*)
call the controller function and return the result

init_controller (*request*)
Get an instance of the controller

Parameters **request** – http request get from pyramid

Return type instance of Pyramid controller

Exception HandlerException

class anyblok_pyramid.handler.**HandlerHTTP** (*namespace*, *view*)
Handler for all PyramidHTTP controllers

wrap_view (*request*)
Call and return the result of wanted controller

Parameters **request** – http request got from pyramid

class anyblok_pyramid.handler.**HandlerRPC** (*namespace*, *method*)
Handler for all PyramidRPC controllers

wrap_view (*request*, **args*, ***kwargs*)
Call and return the result of wanted controller

Parameters

- **request** – http request got from pyramid
- ***args** – list of argument for rpc method

- **`**kwargs`** – list of positional argument for rpc method

3.3 anyblok_pyramid.config

`anyblok_pyramid.config.`**`make_config`**`()`
Return the configuration for pyramid

`anyblok_pyramid.config.`**`declare_static`**`(`*config*`)`
Pyramid includeme, add the static path of the blok

Parameters **`config`** – the pyramid configuration

`anyblok_pyramid.config.`**`pyramid_config`**`(`*config*`)`
Pyramid includeme, add the route and view which are not added in the blok

Parameters **`config`** – the pyramid configuration

`anyblok_pyramid.config.`**`pyramid_http_config`**`(`*config*`)`
Pyramid includeme, add the route and view which are added in the blok by `PyramidHTTP` Type

Parameters **`config`** – the pyramid configuration

Exception PyramidException

`anyblok_pyramid.config.`**`_pyramid_rpc_config`**`(`*cls*, *add_endpoint*, *add_method*`)`
Add the route and view which are added in the blok

Parameters

- **`cls`** – PyramidRPC Type
- **`add_endpoint`** – function to add route in configuation
- **`add_method`** – function to add rpc_method in configuration

Exception PyramidException

`anyblok_pyramid.config.`**`pyramid_jsonrpc_config`**`(`*config*`)`
Pyramid includeme, add the route and view which are added in the blok by `PyramidJsonRPC` Type

Parameters **`config`** – the pyramid configuration

`anyblok_pyramid.config.`**`pyramid_xmlrpc_config`**`(`*config*`)`
Pyramid includeme, add the route and view which are added in the blok by `PyramidXmlRPC` Type

Parameters **`config`** – the pyramid configuration

3.4 anyblok_pyramid.scripts module

`anyblok_pyramid.scripts.`**`anyblok_wsgi`**`(`*description*, *version*, *argsparse_groups*, *parts_to_load*, *Configurator=<function make_config>*`)`

Parameters

- **`description`** – description of argsparse
- **`version`** – version of script for argparse
- **`argsparse_groups`** – list argsparse groupe to load
- **`parts_to_load`** – group of blok to load

- **`Configurator`** – callable which return a config instance

Contents

- *Bloks*
  - `pyramid` *blok*

CHAPTER 4

Bloks

4.1 `pyramid` blok

class `anyblok_pyramid.bloks.pyramid.`**`Pyramid`**(*registry*)

Bases: `anyblok.blok.Blok`

classmethod **`import_declaration_module`**()

classmethod **`reload_declaration_module`**(*reload*)

`required` = **['anyblok-core']**

`version` = **'0.2.0'**

Contents

- *Helper for unittest*
  - *PyramidTestCase*
  - *PyramidDBTestCase*
  - *PyramidBlokTestCase*

CHAPTER 5

Helper for unittest

For unittest, classes are available to offer some fonctionnalities

5.1 PyramidTestCase

```
from anyblok_pyramid.tests.testcase import PyramidTestCase
```

5.2 PyramidDBTestCase

Warning: this testcase destroys the test database for each unittest

5.3 PyramidBlokTestCase

Contents

- *ROADMAP*
  - *To implement*
  - *Functionnality which need a sprint*

CHAPTER 6

ROADMAP

6.1 To implement

- WebSocket
- Add check properties type

6.2 Functionnality which need a sprint

- Internalization

Contents

- *CHANGELOG*
  - *Future*
  - *0.1.0*

CHANGELOG

7.1 Future

- [ADD] configurator callable
- [REF] Adapt the import of python module of the blok, cause of the change in AnyBlok version 0.2.2

7.2 0.1.0

Main version of AnyBlok / Pyramid. You can with this version

- Declare Views / Routes for application
- **Declare controller (Views / Routes) which depend of the installation of bloks**
    - XHR
    - JsonRPC
    - XmlRPC
- Possibility to check some property as authentification
- Possibility to define properties check

Contents

- *Mozilla Public License Version 2.0*
  - *1. Definitions*
    - *1.1. "Contributor"*
    - *1.2. "Contributor Version"*
    - *1.3. "Contribution"*
    - *1.4. "Covered Software"*
    - *1.5. "Incompatible With Secondary Licenses"*
    - *1.6. "Executable Form"*
    - *1.7. "Larger Work"*
    - *1.8. "License"*
    - *1.9. "Licensable"*
    - *1.10. "Modifications"*
    - *1.11. "Patent Claims" of a Contributor*
    - *1.12. "Secondary License"*
    - *1.13. "Source Code Form"*
    - *1.14. "You" (or "Your")*
  - *2. License Grants and Conditions*
    - *2.1. Grants*
    - *2.2. Effective Date*
    - *2.3. Limitations on Grant Scope*
    - *2.4. Subsequent Licenses*
    - *2.5. Representation*
    - *2.6. Fair Use*
    - *2.7. Conditions*
  - *3. Responsibilities*
    - *3.1. Distribution of Source Form*
    - *3.2. Distribution of Executable Form*
    - *3.3. Distribution of a Larger Work*
    - *3.4. Notices*
    - *3.5. Application of Additional Terms*
  - *4. Inability to Comply Due to Statute or Regulation*
  - *5. Termination*
    - *5.1.*
    - *5.2.*
    - *5.3.*
  - *6. Disclaimer of Warranty*
  - *7. Limitation of Liability*
  - *8. Litigation*
  - *9. Miscellaneous*
  - *10. Versions of the License*
    - *10.1. New Versions*
    - *10.2. Effect of New Versions*
    - *10.3. Modified Versions*
    - *10.4. Distributing Source Code Form that is Incompatible With Secondary Licenses*
  - *Exhibit A - Source Code Form License Notice*
  - *Exhibit B - "Incompatible With Secondary Licenses" Notice*

CHAPTER 8

Mozilla Public License Version 2.0

8.1 1. Definitions

8.1.1 1.1. "Contributor"

Means each individual or legal entity that creates, contributes to the creation of, or owns Covered Software.

8.1.2 1.2. "Contributor Version"

Means the combination of the Contributions of others (if any) used by a Contributor and that particular Contributor's Contribution.

8.1.3 1.3. "Contribution"

Means Covered Software of a particular Contributor.

8.1.4 1.4. "Covered Software"

Means Source Code Form to which the initial Contributor has attached the notice in Exhibit A, the Executable Form of such Source Code Form, and Modifications of such Source Code Form, in each case including portions thereof.

8.1.5 1.5. "Incompatible With Secondary Licenses"

Means:

- **That the initial Contributor has attached the notice described in Exhibit B** to the Covered Software; or
- **That the Covered Software was made available under the terms of version 1.1** or earlier of the License, but not also under the terms of a Secondary License.

8.1.6 1.6. "Executable Form"

Means any form of the work other than Source Code Form.

8.1.7 1.7. “Larger Work”

Means a work that combines Covered Software with other material, in a separate file or files, that is not Covered Software.

8.1.8 1.8. “License”

Means this document.

8.1.9 1.9. “Licensable”

Means having the right to grant, to the maximum extent possible, whether at the time of the initial grant or subsequently, any and all of the rights conveyed by this License.

8.1.10 1.10. “Modifications”

Means any of the following:

- **Any file in Source Code Form that results from an addition to, deletion from,** or modification of the contents of Covered Software; or
- Any new file in Source Code Form that contains any Covered Software.

8.1.11 1.11. “Patent Claims” of a Contributor

Means any patent claim(s), including without limitation, method, process, and apparatus claims, in any patent Licensable by such Contributor that would be infringed, but for the grant of the License, by the making, using, selling, offering for sale, having made, import, or transfer of either its Contributions or its Contributor Version.

8.1.12 1.12. “Secondary License”

Means either the GNU General Public License, Version 2.0, the GNU Lesser General Public License, Version 2.1, the GNU Affero General Public License, Version 3.0, or any later versions of those licenses.

8.1.13 1.13. “Source Code Form”

Means the form of the work preferred for making modifications.

8.1.14 1.14. “You” (or “Your”)

Means an individual or a legal entity exercising rights under this License. For legal entities, “You” includes any entity that controls, is controlled by, or is under common control with You. For purposes of this definition, “control” means (a) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (b) ownership of more than fifty percent (50%) of the outstanding shares or beneficial ownership of such entity.

8.2 2. License Grants and Conditions

8.2.1 2.1. Grants

Each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license:

- **Under intellectual property rights (other than patent or trademark)** Licensable by such Contributor to use, reproduce, make available, modify, display, perform, distribute, and otherwise exploit its Contributions, either on an unmodified basis, with Modifications, or as part of a Larger Work; and
- **Under Patent Claims of such Contributor to make, use, sell, offer for sale,** have made, import, and otherwise transfer either its Contributions or its Contributor Version.

8.2.2 2.2. Effective Date

The licenses granted in Section 2.1 with respect to any Contribution become effective for each Contribution on the date the Contributor first distributes such Contribution.

8.2.3 2.3. Limitations on Grant Scope

The licenses granted in this Section 2 are the only rights granted under this License. No additional rights or licenses will be implied from the distribution or licensing of Covered Software under this License. Notwithstanding Section 2.1(b) above, no patent license is granted by a Contributor:

- For any code that a Contributor has removed from Covered Software; or
- **For infringements caused by: (i) Your and any other third party's** modifications of Covered Software, or (ii) the combination of its Contributions with other software (except as part of its Contributor Version); or
- **Under Patent Claims infringed by Covered Software in the absence of its** Contributions.

This License does not grant any rights in the trademarks, service marks, or logos of any Contributor (except as may be necessary to comply with the notice requirements in Section 3.4).

8.2.4 2.4. Subsequent Licenses

No Contributor makes additional grants as a result of Your choice to distribute the Covered Software under a subsequent version of this License (see Section 10.2) or under the terms of a Secondary License (if permitted under the terms of Section 3.3).

8.2.5 2.5. Representation

Each Contributor represents that the Contributor believes its Contributions are its original creation(s) or it has sufficient rights to grant the rights to its Contributions conveyed by this License.

8.2.6 2.6. Fair Use

This License is not intended to limit any rights You have under applicable copyright doctrines of fair use, fair dealing, or other equivalents.

8.2.7 2.7. Conditions

Sections 3.1, 3.2, 3.3, and 3.4 are conditions of the licenses granted in Section 2.1.

8.3 3. Responsibilities

8.3.1 3.1. Distribution of Source Form

All distribution of Covered Software in Source Code Form, including any Modifications that You create or to which You contribute, must be under the terms of this License. You must inform recipients that the Source Code Form of the Covered Software is governed by the terms of this License, and how they can obtain a copy of this License. You may not attempt to alter or restrict the recipients' rights in the Source Code Form.

8.3.2 3.2. Distribution of Executable Form

If You distribute Covered Software in Executable Form then:

- **Such Covered Software must also be made available in Source Code Form, as** described in Section 3.1, and You must inform recipients of the Executable Form how they can obtain a copy of such Source Code Form by reasonable means in a timely manner, at a charge no more than the cost of distribution to the recipient; and
- **You may distribute such Executable Form under the terms of this License, or** sublicense it under different terms, provided that the license for the Executable Form does not attempt to limit or alter the recipients' rights in the Source Code Form under this License.

8.3.3 3.3. Distribution of a Larger Work

You may create and distribute a Larger Work under terms of Your choice, provided that You also comply with the requirements of this License for the Covered Software. If the Larger Work is a combination of Covered Software with a work governed by one or more Secondary Licenses, and the Covered Software is not Incompatible With Secondary Licenses, this License permits You to additionally distribute such Covered Software under the terms of such Secondary License(s), so that the recipient of the Larger Work may, at their option, further distribute the Covered Software under the terms of either this License or such Secondary License(s).

8.3.4 3.4. Notices

You may not remove or alter the substance of any license notices (including copyright notices, patent notices, disclaimers of warranty, or limitations of liability) contained within the Source Code Form of the Covered Software, except that You may alter any license notices to the extent required to remedy known factual inaccuracies.

8.3.5 3.5. Application of Additional Terms

You may choose to offer, and to charge a fee for, warranty, support, indemnity or liability obligations to one or more recipients of Covered Software. However, You may do so only on Your own behalf, and not on behalf of any Contributor. You must make it absolutely clear that any such warranty, support, indemnity, or liability obligation is offered by You alone, and You hereby agree to indemnify every Contributor for any liability incurred by such Contributor as a result of warranty, support, indemnity or liability terms You offer. You may include additional disclaimers of warranty and limitations of liability specific to any jurisdiction.

8.4 4. Inability to Comply Due to Statute or Regulation

If it is impossible for You to comply with any of the terms of this License with respect to some or all of the Covered Software due to statute, judicial order, or regulation then You must: (a) comply with the terms of this License to the maximum extent possible; and (b) describe the limitations and the code they affect. Such description must be placed in a text file included with all distributions of the Covered Software under this License. Except to the extent prohibited by statute or regulation, such description must be sufficiently detailed for a recipient of ordinary skill to be able to understand it.

8.5 5. Termination

8.5.1 5.1.

The rights granted under this License will terminate automatically if You fail to comply with any of its terms. However, if You become compliant, then the rights granted under this License from a particular Contributor are reinstated (a) provisionally, unless and until such Contributor explicitly and finally terminates Your grants, and (b) on an ongoing basis, if such Contributor fails to notify You of the non-compliance by some reasonable means prior to 60 days after You have come back into compliance. Moreover, Your grants from a particular Contributor are reinstated on an ongoing basis if such Contributor notifies You of the non-compliance by some reasonable means, this is the first time You have received notice of non-compliance with this License from such Contributor, and You become compliant prior to 30 days after Your receipt of the notice.

8.5.2 5.2.

If You initiate litigation against any entity by asserting a patent infringement claim (excluding declaratory judgment actions, counter-claims, and cross-claims) alleging that a Contributor Version directly or indirectly infringes any patent, then the rights granted to You by any and all Contributors for the Covered Software under Section 2.1 of this License shall terminate.

8.5.3 5.3.

In the event of termination under Sections 5.1 or 5.2 above, all end user license agreements (excluding distributors and resellers) which have been validly granted by You or Your distributors under this License prior to termination shall survive termination.

8.6 6. Disclaimer of Warranty

Warning: Covered Software is provided under this License on an "as is" basis, without warranty of any kind, either expressed, implied, or statutory, including, without limitation, warranties that the Covered Software is free of defects, merchantable, fit for a particular purpose or non-infringing. The entire risk as to the quality and performance of the Covered Software is with You. Should any Covered Software prove defective in any respect, You (not any Contributor) assume the cost of any necessary servicing, repair, or correction. This disclaimer of warranty constitutes an essential part of this License. No use of any Covered Software is authorized under this License except under this disclaimer.

8.7 7. Limitation of Liability

Warning: Under no circumstances and under no legal theory, whether tort (including negligence), contract, or otherwise, shall any Contributor, or anyone who distributes Covered Software as permitted above, be liable to You for any direct, indirect, special, incidental, or consequential damages of any character including, without limitation, damages for lost profits, loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses, even if such party shall have been informed of the possibility of such damages. This limitation of liability shall not apply to liability for death or personal injury resulting from such party's negligence to the extent applicable law prohibits such limitation. Some jurisdictions do not allow the exclusion or limitation of incidental or consequential damages, so this exclusion and limitation may not apply to You.

8.8 8. Litigation

Any litigation relating to this License may be brought only in the courts of a jurisdiction where the defendant maintains its principal place of business and such litigation shall be governed by laws of that jurisdiction, without reference to its conflict-of-law provisions. Nothing in this Section shall prevent a party's ability to bring cross-claims or counter-claims.

8.9 9. Miscellaneous

This License represents the complete agreement concerning the subject matter hereof. If any provision of this License is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. Any law or regulation which provides that the language of a contract shall be construed against the drafter shall not be used to construe this License against a Contributor.

8.10 10. Versions of the License

8.10.1 10.1. New Versions

Mozilla Foundation is the license steward. Except as provided in Section 10.3, no one other than the license steward has the right to modify or publish new versions of this License. Each version will be given a distinguishing version number.

8.10.2 10.2. Effect of New Versions

You may distribute the Covered Software under the terms of the version of the License under which You originally received the Covered Software, or under the terms of any subsequent version published by the license steward.

8.10.3 10.3. Modified Versions

If you create software not governed by this License, and you want to create a new license for such software, you may create and use a modified version of this License if you rename the license and remove any references to the name of the license steward (except to note that such modified license differs from this License).

8.10.4 10.4. Distributing Source Code Form that is Incompatible With Secondary Licenses

If You choose to distribute Source Code Form that is Incompatible With Secondary Licenses under the terms of this version of the License, the notice described in Exhibit B of this License must be attached.

8.11 Exhibit A - Source Code Form License Notice

```
This Source Code Form is subject to the terms of the Mozilla Public
License, v. 2.0. If a copy of the MPL was not distributed with this file,
You can obtain one at http://mozilla.org/MPL/2.0/.
```

If it is not possible or desirable to put the notice in a particular file, then You may include the notice in a location (such as a LICENSE file in a relevant directory) where a recipient would be likely to look for such a notice.

Note: You may add additional accurate notices of copyright ownership.

8.12 Exhibit B - "Incompatible With Secondary Licenses" Notice

```
This Source Code Form is "Incompatible With Secondary Licenses", as defined
by the Mozilla Public License, v. 2.0.
```

CHAPTER 9

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

anyblok_pyramid.bloks.pyramid, 27
anyblok_pyramid.config, 25
anyblok_pyramid.controllers, 15
anyblok_pyramid.handler, 24
anyblok_pyramid.scripts, 25

A

anyblok_pyramid.bloks.pyramid (module), 27
anyblok_pyramid.config (module), 25
anyblok_pyramid.controllers (module), 15
anyblok_pyramid.handler (module), 24
anyblok_pyramid.scripts (module), 25