pyextdirect Documentation

Release 0.2

Antoine Bertin

September 15, 2012

CONTENTS

Release v0.2

Python implementation of Ext Direct Server-side Stack

CHAPTER

ONE

EXAMPLE

For this simple example we are going to consider that you have knowledge of the client part.

First, in `config.py`:

```
from pyextdirect.configuration import create_configuration, expose

Base = create_configuration()

class Test(Base):
    @expose
    def upper(self, string):
        return string.upper()

@expose(base=Base, action='Test')
def lower(string):
    return string.lower()
```

Then, `api.py`:

```
from pyextdirect.api import create_api
import config

if __name__ == '__main__':
    print "Content-type: text/javascript\r\n\r\n"
    print create_api(config.Base)
```

And finally `router.py`:

```
from pyextdirect.router import Router
import config
import cgi

if __name__ == '__main__':
    router = Router(config.Base)
    fs = cgi.FieldStorage()
    print "Content-type: application/json\r\n\r\n"
    print router.route(fs.value)
```

CHAPTER

TWO

API DOCUMENTATION

2.1 Configuration

pyextdirect.configuration.**BASIC = 0**
Basic method

pyextdirect.configuration.**LOAD = 1**
DirectLoad method

pyextdirect.configuration.**SUBMIT = 2**
DirectSubmit method

pyextdirect.configuration.**STORE_READ = 3**
DirectStore read method

class pyextdirect.configuration.**ConfigurationMeta**(*name*, *bases*, *attrs*)
Each class created with this metaclass will have its exposed methods registered

A method can be exposed with the `expose()` decorator The registration is done by calling `register()`

pyextdirect.configuration.**create_configuration**(*name='Base'*)
Create a configuration base class

It is built using `ConfigurationMeta`. Subclassing such a base class will register exposed methods

class pyextdirect.configuration.**Base**

configuration
Configuration dict that can be used by a Router or the API

classmethod register(*element*, *action*, *method*)
Register an element in the `configuration`

Parameters
- **element** (*tuple of (class, method name) or function*) – the element to register
- **action** (*string*) – name of the exposed action that will hold the method
- **method** (*string*) – name of the exposed method

pyextdirect.configuration.**expose**(*f=None*, *base=None*, *action=None*, *method=None*, *kind=0*)
Decorator to expose a function

Note: A module function can be decorated but `base` parameter has to be specified

---

Parameters

- **f** (*function or None*) – function to expose
- **base** – base class that can register the function
- **action** (*string*) – name of the exposed action that will hold the method
- **method** (*string*) – name of the exposed method
- **kind** (`BASIC` or `LOAD` or `SUBMIT`) – kind of the method

2.2 Router

class `pyextdirect.router.`**`Router`**(*bases*)

Router able to route Ext.Direct requests to their right functions and provide the result

Parameters **bases** (`Base` or list of `Base`) – configuration bases

`call`(*request*)

Call the appropriate function

Parameters **request** (*dict*) – request that describes the function to call

Returns response linked to the request that holds the value returned by the called function

Return type dict

`route`(*data*)

Route an Ext Direct request to the appropriate function(s) and provide the response(s)

Parameters **data** (*json or dict or list*) – Ext Direct request

Returns appropriate response(s)

Return type json

`pyextdirect.router.`**`create_instances`**(*configuration*)

Create necessary class instances from a configuration with no argument to the constructor

Parameters **configuration** (*dict*) – configuration dict like in `configuration`

Returns a class-instance mapping

Return type dict

2.3 API

`pyextdirect.api.`**`create_api`**(*bases*, *url*, ***kwargs*)

Create the JS code for the API using one or more `Bases`

Parameters

- **bases** (`Base` or list of `Base`) – configuration bases
- **url** (*string*) – see `create_api_dict()`
- ****kwargs** – see `create_api_dict()`

`pyextdirect.api.`**`create_api_dict`**(*bases*, *url*, ***kwargs*)

Create an API dict

Parameters

- **bases** (`Base` or list of `Base`) – configuration bases

- **url** (*string*) – URL where the router can be reached
- ****kwargs** – extra keyword arguments to populate the API dict. Most common keyword arguments are *id*, *maxRetries*, *namespace*, *priority* and *timeout*

Note: Keyword arguments *type*, *url*, *actions* and *enableUrlEncode* will be overridden

---

2.4 Exceptions

exception `pyextdirect.exceptions.`**`Error`**
Base class for pyextdirect exceptions

exception `pyextdirect.exceptions.`**`FormError`**(*errors=None*, *extra=None*)
Raised when an error occurs in a form handler method

A result dict can be provided and will be returned to the client

Parameters

- **errors** (*dict*) – result returned to the client
- **extra** (*dict*) – extra stuff in the returned result

For example:

```
@expose(kind=SUBMIT)
def save(self, firstname, lastname):
    errors = {}
    if len(firstname) < 3:
        errors['firstname'] = 'Invalid length'
    if not lastname:
        errors['lastname'] = 'Missing'
    if errors:
        raise FormError(errors, {'foo': 'bar'})
    # ...
```

Will return the dict `{'success': False, 'errors': {'firstname': 'Invalid length', 'lastname': 'Missing'}, 'foo': 'bar'}` as result in a response

PYTHON MODULE INDEX

p

pyextdirect.api, **??**
pyextdirect.configuration, **??**
pyextdirect.exceptions, **??**
pyextdirect.router, **??**