Apuntes web2py

Sergio Alvariño salvari@gmail.com

abril-2017

Resumen

Apuntes de web2py
Solo para referencia rápida y personal.

Índice general

1 Web2py

Toda la info aquí

2 Tutoriales en la red

2.1 Killer Web Development por Marco Laspe

Disponible aquí

2.1.1 Cambios

`Crud` debe ser importado explícitamente:

```
from gluon.tools import Crud
crud = Crud(db)
```

Despues ya podemos seguir el tutorial haciendo por ejemplo:

```
def entry_post():
    """returns a form where the user can entry a post"""
    form = crud.create(db.post)
    return dict(form=form)
```

3 Python

3.1 Decorators

Decorators añadidos desde Python 2.4 para permitir que el *function and method wrapping* fuese más fácil de leer y entender. *function and method wrapping* consiste en implementar una función (o método) que recibe como parámetro una función (¿o método?) y devuelve una función mejorada.

El caso de uso original era definir los métodos como métodos de Clase o métodos estáticos en la cabecera de su definición.

La recete general:

```
def mydecorator(function):
    def _mydecorator(*args, **kw):
        # do some stuff before the real
        # function gets called
        res = function(*args, **kw)
        # do some stuff after
        return res
    # returns the sub-function
    return _mydecorator
```

El intérprete carga los *decorators* cuando se lee el módulo la primera vez, debe limitarse su uso a *wrappers* que puedan aplicarse de forma genérica. Si el *decorator* está fuertemente acoplado con la clase o función que decora debería reescribirse y convertirlo en un invocable regular para evitar la complejidad.

Patrónes típicos:

- Argument checking
- Caching
- Proxy
- Context provider

Nota: En web2py parece que los *decorators* se usan tipicamente como proveedores de contexto. Hay que ver como funciona la sentencia `with` de Python 2.5 que se crea con el mismo propósito.