Nagare – Pyconfr 2011

Hervé Coatanhay – Net-ng

herve.coatanhay@net-ng.com

流れ

Différent

Frameworks MVC	Nagare
Java, XML, SQL, HTML, JSP, JS, langage de template	Python
Routage d'URL	Association de Callback
Multiplication de « controllers »	Programmation linéaire & Continuation
État stocké manuellement en session	Sérialisation automatique du graphe de composant
Gestion manuelle des « back » et des « fork »	Gestion automatique

Mais…

- WSGI
- SQLAlchemy / Elixir
- Lxml
- Paste / Flup
- Python-memcached
- WebError / WebOb
- Stackless Python
- Pyjamas Py2Js
- Setuptools
- PEAK Rules
- ConfigObj
- VirtualEnv
- YUI
- Nose

Exemple

```
class TODOList(object):
    def __init__(self):
        self.items = [u'Acheter de la bière',
                      u'Prendre un café',
                      u'Jouer à Angry Birds',
                      u'Prendre un autre café',
                      u'Acheter un iPhone5 !']


@presentation.render_for(TODOList)
def render(self, h, *args):
    with h.ol:
        for item in self.items:
            h << h.li(h.a(item))
    return h.root
```

Action

```
class TODOList(object):
    def __init__(self):
        self.items = [[u'Acheter de la bière', False],
                      [u'Prendre un café', False],
                      [u'Jouer à Angry Birds', False],
                      [u'Prendre un autre café', False],
                      [u'Acheter un iPhone5 !', False]]

    def toggle_item(self, index):
        self.items[index][1] = not self.items[index][1]

@presentation.render_for(TODOList)
def render(self, h, *args):
    h.head.css("done", "li.done {border: 1px solid red}")

    with h.ol:
        for (i, (item, done)) in enumerate(self.items):
            h << h.li(
                      h.a(item).action(lambda i=i:self.toggle_item(i)),
                      class_="done" if done else "todo"
                      )
    return h.root
```

Navigation

```
@presentation.render_for(TODOList)
def render(self, h, comp, *args):
    h.head.css("done", "li.done {border: 1px solid red}")
    with h.ol:
        for (i, (item, done)) in enumerate(self.items):
            h << h.li(
                    h.a(item).action(lambda i=i:self.toggle_item(i)),
                    class_="done" if done else "todo"
                    )
    h << h.a(u'filtrer').action(lambda: comp.becomes(self, 'filtered'))
    return h.root
```

Formulaire

```
@presentation.render_for(TODOList)
def render(self, h, comp, *args):
    h.head.css("done", "li.done {border: 1px solid red}")

    with h.ol:
        for (i, (item, done)) in enumerate(self.items):
            h << h.li(
                    h.a(item).action(lambda i=i:self.toggle_item(i)),
                    class_="done" if done else "todo"
                    )
    h << component.Component(self, model='add_form')
    return h.root
```

```
@presentation.render_for(TODOList, model='add_form')
def render(self, h, comp, *args):
    with h.form:
        h << h.input(type="text", value="").action(self.add_item)
    return h.root
```

Réutilisation

```
class MyApp(object):

    def __init__(self):
        self.list1 = component.Component(TODOList())
        self.list2 = component.Component(TODOList())


@presentation.render_for(MyApp)
def render(self, h, comp, *args):
    with h.ul:
        h << h.li(self.list1)
        h << h.li(self.list2)
    return h.root
```

En asynchrone

```
class MyApp(object):

    def __init__(self):
        self.list1 = component.Component(TODOList())
        self.list2 = component.Component(TODOList())
        self.nb_display = 0


@presentation.render_for(MyApp)
def render(self, h, comp, *args):
    self.nb_display += 1

    with h.ul:
        h << h.AsyncRenderer().li(self.list1)
        h << h.li(self.list2)
    h << h.div(self.nb_display)

    return h.root
```

Persistance

```
class TODOListData(Entity):
    name = Field(String(255), default='', unique=True, index=True, nullable=False)
    items = OneToMany('TODOListItemData')

class TODOListItemData(Entity):
    title = Field(String(255))
    done = Field(Boolean, default=False)

    todo_list = ManyToOne('TODOListData')
```

URL significative

```
@presentation.init_for(TODOList, "(len(url) == 1)")
def init(self, url, *args):
    self.list_name = url[0]
```

Questions ?

www.nagare.org

Google groups: nagare-user
IRC: Freenode #nagare
Twitter: nagareproject