La GUI di Genropy

Chi siamo

- Giovanni Porcari:
  Padre del Framework Genropy sviluppatore dal 73
- Francesco Porcari:
  sviluppa in Python dal 2006 e ha un amore nascosto per JavaScript

Per informazioni: www.genropy.org

Genropy è un framework per costruire applicazioni gestionali

Ma non solo...

Genropy è una GUI per il WEB

- Un modo per scrivere interfacce grafiche complesse usando solo Python
- Una metodologia MVC che offre il modo di rafforzare la separazione tra i dati e la loro rappresentazione
- Un modo semplice e veloce per eseguire dei compiti sul server e presentare dei risultati

Ma come funziona ?

- Capire la logica di una GUI è fondamentale per comprendere quello che ci possiamo fare
- Genropy ha un suo specifico modo di gestire l'interfaccia nell'applicazione
- E ora lo vediamo...

HTML no grazie

- HTML è un modo di mandare al client una descrizione di come comporre la pagina. Ma da scrivere è assai poco piacevole.
- Con il codice HTML noi riusciamo a descrivere il DOM che il browser dovrà costruire.
- I framework web classici hanno lo scopo di creare codice HTML.

Il DOM da Javascript

- In Genropy il DOM è generato da un builder a partire da una ricetta.

- Non questo tipo di ricetta però!

La ricetta della pagina

- La ricetta di una pagina è costituita da una Bag, ovvero da un contenitore gerarchico, che si popola con chiamate python.
- La nostra ricetta viene poi serializzata in XML e spedita al client.
- Il client deserializza la ricetta e crea una Bag sul client che è l'esatta immagine di quella preparata sul server.

Che diavolo è una Bag ?

- Una Bag è un contenitore gerarchico con delle API uguali in Python e Javascript.
- Una Bag è una collezioni di Bagnode.
- Un Bagnode è un oggetto che ha una label, un valore e degli attributi.
- Una Bag può contenuti lazy e cached
- Una Bag può reagire ad eventi (trigger)

Come funziona una pagina di Genropy

- L'utente esegue una chiamata ad un URL
- Viene resa una pagina HTML minimale
- Viene istanziato un client Javascript (genro)
- Il client fa una chiamata per farsi dare la ricetta di quella pagina.
- Il server rende la ricetta che viene salvata nello store (Bag) chiamato Source
- Il builder, attivato dall'arrivo della ricetta, costruisce la GUI in base alla ricetta.

Esempio Hello World

```
class GnrCustomWebPage(object):
    def main(self,root,**kwargs):
        root.h1('Hello world',text_align='center')
```

- Una pagina è un modulo python dove è definita una classe GnrCustomWebPage.
- La classe eredita da object e quindi è solo un raccoglitore di metodi.

Esempio Hello World (segue)

- La classe può definire un metodo **main** che riceve il nodo di partenza della ricetta (root).
- Al nodo aggiungiamo un elemento di tipo h1 con un valore ‘Hello world’ e un attributo css text_align=‘center’

Hello World : la ricetta XML

```
<GenRoBag
    <result>
        <h1_1 _T="BAG" tag="h1"
              text_align="center"
              innerHTML="Hello world"/>
    </result>
</GenRoBag>
```

La ricetta è viva

- La bag con il Source è vigilata e ad ogni cambiamento viene invocato il builder per aggiornare il DOM.
- Se dall'inspector cambiamo dei valori o attributi nella bag il contenuto della pagina si adegua.

I dati nella pagina

- Oltre al contenitore del Source la pagina ha un contenitore per i dati : Data.
- Il contenitore dei dati fornisce la parte variabile ovvero il contenuto.
- Possiamo inizializzare lo store dei dati con l'istruzione **.data**.
- Possiamo usare un valore dello store con l'operatore '^'

Testo preso dallo store

```
class GnrCustomWebPage(object):
    def main(self,root,**kwargs):
        root.data('foo','Hello world')
        root.h1('^foo',text_align='center')
```

- Il contenitore dei dati viene precaricato con “Hello world” all’indirizzo “foo”
- L’elemento h1 tramite l’operatore ‘^’ trova il suo valore all’indirizzo “foo”

Modifica dati store

```
class GnrCustomWebPage(object):
    def main(self,root,**kwargs):
        root.data('foo','Hello world')
        root.h1('^foo',text_align='center')
        root.textbox('^foo',margin_top='20px')
```

- Aggiungiamo un widget di input per modificare il dato all'indirizzo 'foo'

Possiamo modificare solo il valore ?

- Risposta breve : No.
- Ogni parametro può essere modificato.
- Per rendere l'esempio più usabile introduciamo il **formbuilder**.
- Il **formbuilder** ci permette di impaginare bene i valori da riempire assegnando anche un'etichetta.

Possiamo modificare solo il valore ?

```
class GnrCustomWebPage(object):
    def main(self,root,**kwargs):
        box = root.div(border='1px solid silver',
                       margin='5px',padding='5px')
        fb = box.formbuilder(border_spacing='3px')
        fb.textbox('^mytext',lbl='Text')
        fb.textbox('^mycolor',lbl='Color')
        fb.textbox('^myfontsize',lbl='Font size')
        fb.div('^mytext',color='^mycolor',text_align='center',
               font_size='^myfontsize',lbl='Result')
```

Ora ne facciamo 3

...ma resteremo delusi

```
class GnrCustomWebPage(object):
    def main(self,root,**kwargs):
        for k in range(3):
            self.sampleblock(root)

    def sampleblock(self,pane):
        box = pane.div(border='1px solid silver',
                       margin='5px',padding='5px')
        fb = box.formbuilder(border_spacing='3px')
        fb.textbox('^mytext',lbl='Text')
        fb.textbox('^mycolor',lbl='Color')
        fb.textbox('^myfontsize',lbl='Font size')
        fb.div('^mytext',color='^mycolor',text_align='center',
               font_size='^myfontsize',lbl='Result')
```

Ora ne facciamo 3

...e grazie ai path relativi tutto funziona come deve

```
class GnrCustomWebPage(object):
    def main(self,root,**kwargs):
        for k in range(3):
            self.sampleblock(root.div(datapath='block_%02i' %k))

    def sampleblock(self,pane):
        box = pane.div(border='1px solid silver',
                       margin='5px',padding='5px')
        fb = box.formbuilder(border_spacing='3px')
        fb.textbox('^.mytext',lbl='Text')
        fb.textbox('^.mycolor',lbl='Color')
        fb.textbox('^.myfontsize',lbl='Font size')
        fb.div('^.mytext',color='^.mycolor',text_align='center',
               font_size='^.myfontsize',lbl='Result')
```

Usiamo sh.py

```
from gnr.core.gnrdecorator import public_method
import sh

class GnrCustomWebPage(object):
    def main(self,root,**kwargs):
        box = root.div(datapath='test')
        fb = box.formbuilder(border_spacing='3px',cols=2)
        fb.textbox(value='^.command',lbl='Command',width='40em')
        box.dataRpc('.result',self.doCommand,
                    command='^.command',_if='command')
        box.pre('^.result')

    @public_method
    def doCommand(self,command=None,parameters=None):
        try:
            cmdlist = command.split(' ')
            command = cmdlist.pop(0)
            cmd = getattr(sh,command)
            return cmd(*cmdlist) if cmdlist else cmd()
        except Exception, e:
            return str(e)
```

Spediamo una email

```
class GnrCustomWebPage(object):
    def main(self,root,**kwargs):
        box = root.div(datapath='mailclient')
        fb = box.formbuilder(border_spacing='3px',cols=1,datapath='.data')
        fb.textbox(value='^.to_address',lbl='To',width='40em')
        fb.textbox(value='^.subject',lbl='Subject',width='40em')
        fb.simpleTextArea(value='^.body',lbl='Body',width='40em',height='80px')
        fb.button('Send',fire='mailclient.send')
        box.dataRpc('.result',self.sendEmail,data='=.data',_fired='^.send')
        box.pre('^.result')

    @public_method
    def sendEmail(self,command=None,data=None):
        try:
            self.site.getService('mail').sendmail_template(data)
            return "Successfully sent email"
        except Exception:
            return "Error: unable to send email"
```

Spediamo una email async

```
@public_method
def sendEmail(self,command=None,data=None):
    try:
        mailserver = self.site.getService('mail')
        mailserver.sendmail_template(data,async=True,cb=self.onSentEmail)
        return 'Sending email'
    except Exception:
        return "Error: unable to send email"

def onSentEmail(self):
    from time import sleep
    sleep(3)
    self.setInClientData('mailclient.result','Email sent')
```

Vediamo una directory

```
class GnrCustomWebPage(object):

    def main(self,root,**kwargs):
        bc=root.borderContainer(datapath='diskviewer')
        left=bc.contentPane(region='top',height='50%',splitter=True)
        left.data('.root.genropy',DirectoryResolver(PATH)())
        left.tree(storepath='.root',hideValues=True,
                  selectedLabelClass='selectedTreeNode',
                  selected_abs_path='.abs_path',
                  labelAttribute='nodecaption',autoCollapse=True)
    left.dataRpc('.content',self.getContent, filepath='^.abs_path')
    bc.contentPane(region='center').pre(value='^.content',font_size='.8em')

@public_method
def getContent(self,filepath=None,**kwargs):
    filepath=os.path.join(PATH,filepath)
    with open(filepath,'r') as f:
        data=f.read()
    return data
```

Top processi: interfaccia

```
class GnrCustomWebPage(object):

    def main(self,root,**kwargs):
        properties=['pid','ppid','name','username','status','create_time',
                    'cpu_percent','memory_percent','cwd','nice','uids',
                    'gids','cpu_times','memory_info','exe']
        pane =root.div(margin='15px',datapath='processList')
        fb=pane.formbuilder(cols=1)
        fb.numberTextBox(value='^.treshold',lbl='Treshold',default=.5)
        fb.checkBoxText('^.columns',values=','.join(properties),colspan=2,
                        default='pid,name,username,cpu_percent',
                        cols=4,width='100%',lbl='Columns')
        pane.dataRpc('.data', self.getProcessesBag, columns='^.columns',
                        treshold='^.treshold',_timing=2,_onStart=True)
        pane.quickGrid(value='^.data',height='200px',width='100%',
                         sortedBy='cpu_percent:d',border='1px solid silver')
@public_method
def getProcessesBag(self,columns=None,treshold=None):
    ...
```

Top processi:Rpc

```
class GnrCustomWebPage(object):
    def main(self,root,**kwargs):
        ...
    @public_method
    def getProcessesBag(self,columns=None,treshold=None):
        columns=(columns or 'pid,name').split(',')
        result = Bag()
        for p in psutil.process_iter():
            d=p.as_dict()
            if d['cpu_percent']>(treshold or 0):
                d['create_time'] = datetime.fromtimestamp(d['create_time'])
                d['cpu_percent'] = d['cpu_percent'] or 0.
                d['memory_percent'] = d['memory_percent'] or 0.
                row = Bag([(k,d[k]) for k in columns if k in d])
                result.setItem('p_%s'%p.pid,row)
        return result
```

Domande?