TurboGears, framework de Python per aplicacions web

I Jornades GPL Tarragona
29 de Gener de 2009

Tomàs Reverter Morelló
tomas.reverter@lotomas.net
http://lotomas.net

http://www.gpltarragona.org

GPL TARRAGONA

Llicència Creative Commons BY-SA

Reconeixement-Compartir amb la mateixa llicència 3.0 Espanya

Sou lliure de:

copiar, distribuir i comunicar públicament l'obra

fer-ne obres derivades

Amb les condicions següents:

Reconeixement. Heu de reconèixer els crèdits de l'obra de la manera especificada per l'autor o el llicenciador (però no d'una manera que suggereixi que us donen suport o rebeu suport per l'ús que feu l'obra).

Compartir amb la mateixa llicència. Si altereu o transformeu aquesta obra, o en genereu obres derivades, només podeu distribuir l'obra resultant amb la mateixa llicència, una de similar o una de compatible.

- Quan reutilitzeu o distribuïu l'obra, heu de deixar ben clar els termes de la llicència de l'obra.
- Alguna de les condicions pot no aplicar-se si obteniu el permís del titular dels drets d'autor.
- No hi ha res en aquesta llicència que menyscabi o restringeixi els drets morals de l'autor.

http://creativecommons.org/licenses/by-sa/3.0/es/deed.ca

Taula de contingut

Què és TurboGears?

- TurboGears és un *framework* web basat en Python, un mega-framework
  - Comparable a Ruby On Rails i CakePHP
- Permet construir aplicacions web orientades a bases de dades
- Construït amb arquitectura MVC
- Codi obert (llicència MIT, compatible GPL)
- Multiplataforma
- Projecte amb gairebé 5 anys de vida
  - Versió estable 1.0.8
  - Versió 2.0 en beta 4

I un framework?

- I un *framework*?
  - *A software framework is an abstraction in which common code providing generic functionality can be selectively overridden or specialized by user code providing specific functionality. ... the overall program's flow of control is not dictated by the caller, but by the framework. This inversion of control is the distinguishing feature of software frameworks. [Wikipedia]*
- Un *framework* és una peça de programari sobre el que “pugem” els desenvolupadors per aprofitar funcionalitats comuns
  - “pugem” = Inversion of Control!
- No hem de reinventar la roda!
- A la pràctica = un esquelet en que basar-te

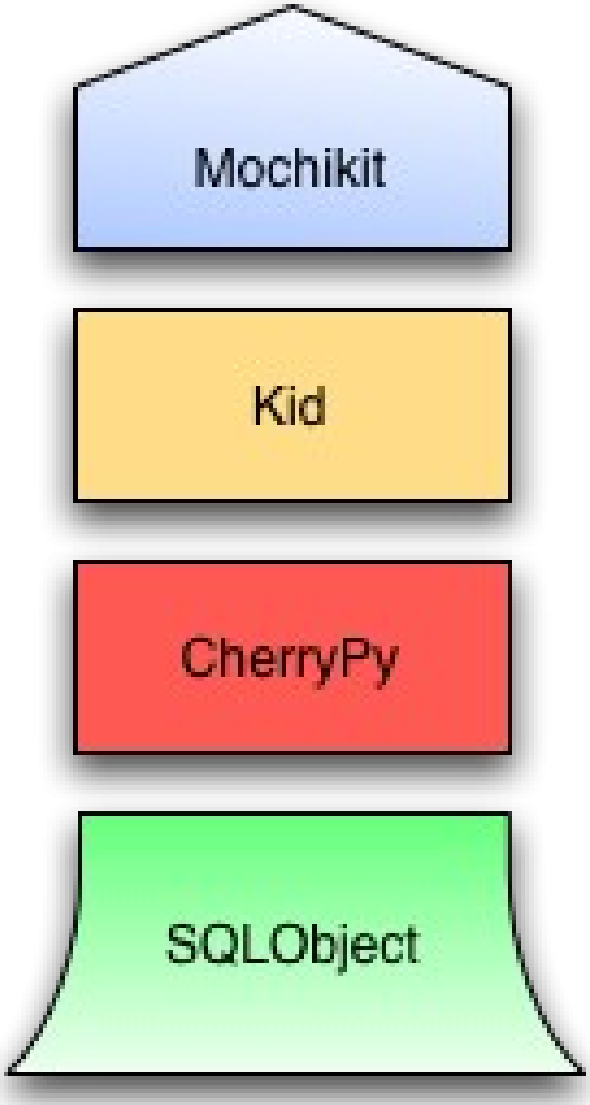
Model-Vista-Control (MVC)

- El MVC és un patró arquitectònic de l'enginyeria del programari
- Busca la separació de components per simplificar les aplicacions
  - Independitza la presentació de les dades
- Parts:
  - Model: s'encarrega de la persistència, la base de dades
  - Vista: la UI del client, la capa de presentació
  - Control: la lògica del servidor

Flux del Model-Vista-Control (MVC)

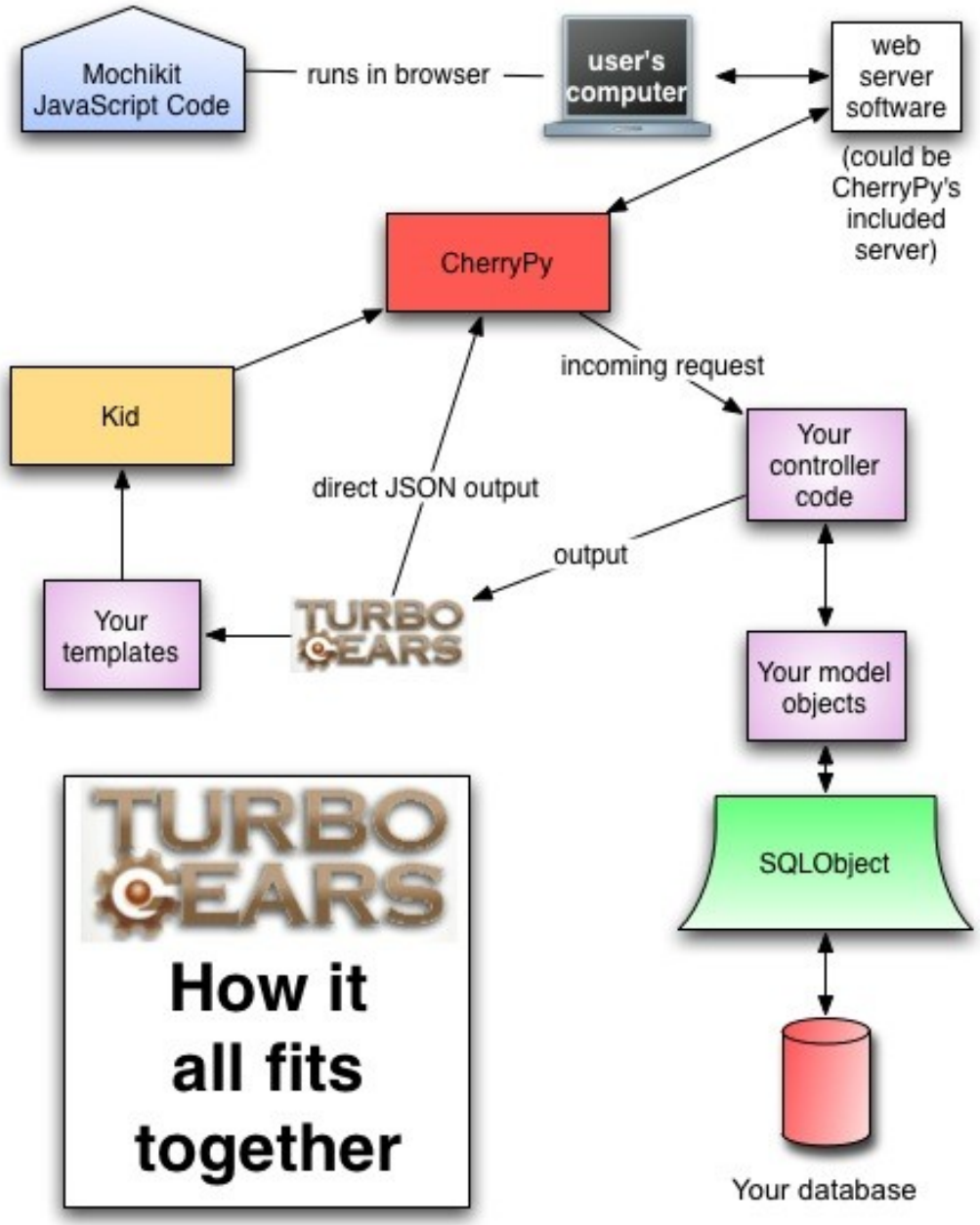
- L'usuari interactua amb la interfície gràfica dissenyada a la vista
- El control manega l'entrada de l'usuari, redirigint la petició a una peça de codi més específica
- El control actualitza la informació del model
- La vista utilitza el model per generar la nova interfície d'usuari
- La interfície resta a l'espera de l'acció de l'usuari

Components, components i components



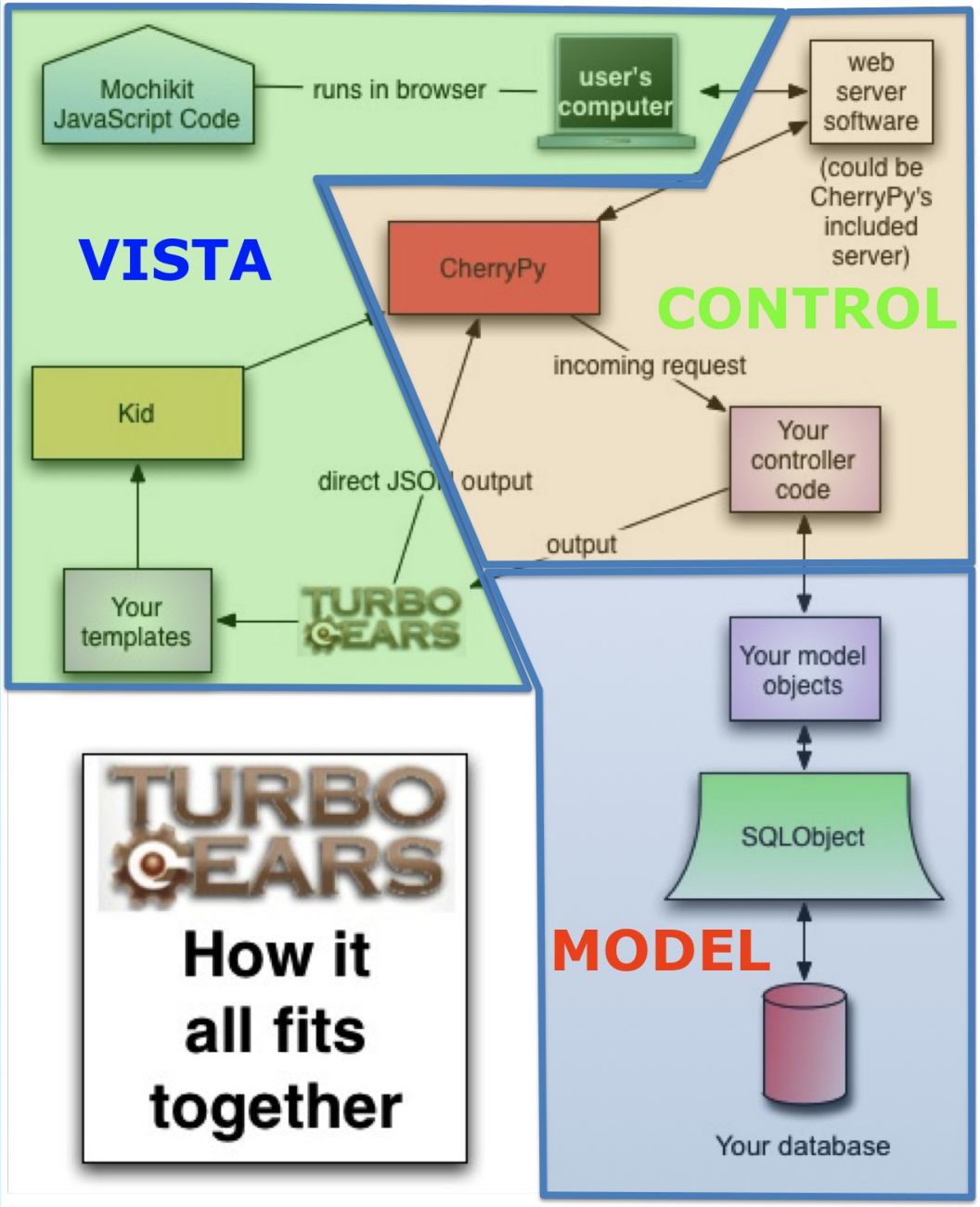
- Client-side JavaScript
  - Mochikit
- Template engine
  - Kid
- Servidor d'aplicacions
  - CherryPy
- Abstracció base de dades (ORM)
  - SQLObject
- I molts més...
  - FormEncode, Nose, simplejson, SQLAlchemy, Genshi, Identity Management...

Model-Vista-Control (MVC) i l'estructura de TurboGears



- Filosofia Jack el Destripador: *"anem per parts"*
- En cada component es poden utilitzar diverses solucions

Model-Vista-Control (MVC) i l'estructura de TurboGears



- Filosofia Jack el Destripador: *"anem per parts"*
- En cada component es poden utilitzar diverses solucions

Hands-on-lab: un wiki en 20 minuts

- Demostració pràctica del funcionament de TurboGears
- Desenvoluparem una aplicació web que ens doni la funcionalitat de wiki:
  - Guardar informació per paraules
  - Fàcil edició de continguts
  - Reconeixement de sintaxis CamelCase (per enllaços)
  - URLs boniques
  - Sortida en XML/JSON
  - Ginys a la interfície en AJAX

Hands-on-lab: requeriments

- Els requeriments per realitzar aquesta demo són:
  - TurboGears 1.0.x
    - GNU/Linux, MacOSX o Windows
    - Python >= 2.3.x, <= 2.5.x (no 2.6 ni 3k)
    - Connexió a internet (easy_install)
  - Base de dades (MySQL, SQLite, PostgreSQL...)
- Opcionalment, un IDE per Python
  - Eclipse
  - EasyEclipse for Python
  - VIM

Hands-on-lab: passos que seguirem

1. Crear l'esquelet de l'aplicació (quickstart)
2. Definir el model de dades
    - Crear la base de dades
    - Escriure dades de mostra
3. Escriure les plantilles de la vista
4. Escriure el controlador en Python
5. Rubricar amb JSON i AJAX

QuickStart: obrint el pastís

- Crear un esquelet de projecte des de zero

```
$ tg-admin quickstart
Enter project name: gplWiki20
Enter package name [wiki20]: gplWiki20
Do you need Identity (usernames/passwords) in this
project? [no] no

...output...

cd gplWiki20
```

- Amb això ja tenim un esquelet d'aplicació web totalment funcional

I no fa falta servidor web?

- Als requeriments, no hi ha servidor web!!!
  - CherryPy ja està integrat amb TurboGears
- Per arrancar l'aplicació:

```
cd gplWiki20

python start-gplwiki20.py
```

- Punt d'entrada -> `http://localhost:8080`
- Tant senzill com això!

Llei del mínim esforç: definint el model via ORM

- Model centralitzat en un fitxer:

  `gplwiki20/model.py`

- Definim la entitat Page:

```
class Page(SQLObject):
    pagename = UnicodeCol(alternateID=True,length=30)
    data = UnicodeCol()
```

- Amb aquesta definició d'SQLObject en tenim prou
- (Vigilar amb els tabuladors, que formen part de la sintaxi del Python!)

Creació de les taules a la base de dades

- Primer hem de configurar la base de dades
  - Dos fitxers de configuració específics, desenvolupament (`dev.cfg`) i per producció (`prod.cfg`)
- Fitxer *dev.cfg*

```
sqlobject.dburi="mysql://us:pwd@localhost/gplWiki20"
```

- Crear la base de dades en si

```
echo "create database gplWiki20" | mysql -u root
```

- Sou SQL-desconfiats? Anem a veure que executarà:

```
tg-admin sql sql
```

- Ara, a crear taules en una comanda:

```
tg-admin sql create
```

Utilizant dummies pels crash-tests

- La informació per proves es pot entrar fàcilment des de la toolboox de TurboGears

  ```
  tg-admin toolbox
  ```

- Ens portarà a http://localhost:7654/
  - Anar a *CatWalk*
  - Clic a *Page*
  - *Add Page* amb el nom FrontPage
- El toolbox pot ajudar per moltes més coses:
  - Informació de TurboGears (paquets, versions...)
  - WidgetBrowser, veure les capacitats dels widgets (!!!!)
  - admin18n, eina d'administració d'strings i18n
  - ModelDesigner, GUI per generar el model.py

Treballant amb templates

- **.kid* i els altres *templates* es poden obrir directament en un navegador!
- Per tant, poden ser editades pels dissenyadors web amb les seves aplicacions super-cool!
- Només s'ha d'anar amb compte de posar el text d'exemple

```
<span py:replace="page.pagename">Text de la pàgina</span>
```

L'aparença també compta (templates)

- Crear el template a partir de l'esquelet

```
cp gplWiki20/templates/welcome.kid \
     gplWiki20/templates/page.kid
```

- Editar *gplwiki20/templates/page.kid* amb:

```
<title> ${page.pagename} - 20 Minute Wiki </title>
...
<div class="main_content">
 <div style="float:right; width: 10em">
   Viewing <span py:replace="page.pagename">Text de
   la pàgina</span>
   <br/>
   You can return to the <a href="/">FrontPage</a>.
 </div>
 <div py:replace="XML(data)">Page text here.</div>
</div>
```

Però la potència sense control no serveix de res

- Escriurem el controlador: *gplwiki20/controllers.py*

```
import turbogears
from turbogears import controllers, expose
from gplwiki20.model import Page
from docutils.core import publish_parts

class Root(controllers.RootController):
    @expose(template="gplwiki20.templates.page")    #1
    def index(self , pagename="FrontPage"):         #2
        page = Page.byPagename(pagename)            #3
        content = publish_parts(page.data,
            writer_name="html")['html_body']        #4
        return dict(data=content, page=page)        #5
```

- Observeu que sense fer res s'ha recarregat el servidor web i els canvis ja seran visibles!

Però la potència sense control no serveix de res – Edició de pàgines

- Crear el template a partir del *page.kid*

```
cp gplWiki20/templates/page.kid \
     gplWiki20/templates/edit.kid
```

- Editar *gplwiki20/templates/edit.kid* amb:

```
<title> Editant ${page.pagename} ... </title>
...
<div class="main_content">
...
 <form action="save" method="post">
   <input type="hidden" name="pagename"
      value="${page.pagename}"/>
   <textarea name="data" py:content="page.data"
      rows="10" cols="60"/>
   <input type="submit" name="submit" value="Save"/>
 </form>
 <!-- Eliminar <div py:replace="XML(data)">Text de la
   pàgina.</div> -->
</div>
```

Però la potència sense control no serveix de res – Edició de pàgines II

- Crear el mètode d'acció a *controllers.py*

```
    @expose("gplwiki20.templates.edit")
    def edit(self, pagename):
        page = Page.byPagename(pagename)
        return dict(page=page)
```

Tab!

- Crear el mètode de guardar a *controllers.py*

```
    @expose()
    def save(self, pagename, data, submit):
        page = Page.byPagename(pagename)
        page.data = data
        turbogears.flash("Changes saved!")
        raise turbogears.redirect("/",pagename=pagename)
```

- Afegir l'enllaç a *page.kid*

```
<p>
  <a href="${tg.url('/edit',pagename=page.pagename)}">
Editar la pàgina</a>
</p>
```

Però la potència sense control no serveix de res – Llistat de pàgines

- Crear el template a partir del *page.kid*

```
cp gplWiki20/templates/page.kid \
     gplWiki20/templates/pagelist.kid
```

- Editar *gplwiki20/templates/pagelist.kid* amb:

```
<title> Llistat </title>
...
<div class="main_content">
  <h1>Totes les pàgines</h1>
  <ul>
    <li py:for="pagename in pages">
      <a href="${tg.url('/' + pagename)}"
        py:content="pagename">Nom de la pàgina.</a>
    </li>
  </ul>
</div>
```

Però la potència sense control no serveix de res – Llistat de pàgines II

- Crear el mètode d'acció a *controllers.py*


```
@expose("gplwiki20.templates.pagelist")
def pagelist(self):
    pages = [page.pagename for page in
             Page.select(orderBy=Page.q.pagename)]
    return dict(pages=pages)
```

- Afegir l'enllaç a *master.kid*

```
...
<div id="footer">
...
<p>Veure la <a href="${tg.url('/pagelist')}">llista
completa de pàgines.</a></p>
...
```

Les URLs entren pels ulls

- Actualment per accedir a les pàgines:

  http://localhost:8080/?pagename=FrontPage

- No és molt bonic no?? Potser millor

  http://localhost:8080/FrontPage

- Definim el mètode per defecte a *controllers.py*


```
@expose()
def default(self, pagename):
    return self.index(pagename)
```

- Al mètode *save* canviem

```
raise turbogears.redirect("/", pagename=pagename)
```

per

```
raise turbogears.redirect("/%s" % pagename)
```

Creant les petjades dels camells

- Senzillament podem tenir sintaxis CamelCase, a través d'expressions regulars (al principi de *controllers.py*)

```
import re
wikiwords = re.compile(r"\b([A-Z]\w+[A-Z]+\w+)")
```

- Modifiquem

```
def index(self, pagename="FrontPage"):
    content = ...
    root = str(turbogears.url('/'))
    content = wikiwords.
       sub(r'<a href="%s\1">\1</a>'
       % root, content)
    return dict(data=content, page=page)
```

I si les coses no van bé?

- Si la pàgina no existeix, la crearem (*controllers.py*):

```
@expose("gplwiki20.templates.edit")
def notfound(self, pagename):
    page = Page(pagename=pagename, data="")
    return dict(page=page)
```

- Si la pàgina no existeix, la crearem (*controllers.py*):

```
@expose(template="gplwiki20.templates.page")
def index(self, pagename="FrontPage"):
    try:
        page = Page.byPagename(pagename)
    except SQLObjectNotFound:
        raise turbogears.redirect("notfound"
                                  , pagename = pagename)
...
```

- I un header per incloure l'excepció

```
from sqlobject import SQLObjectNotFound
```

Necessitem ser oberts i comunicar-nos

- Podem generar informació a XML, JSON...

```
@expose("gplwiki20.templates.pagelist")
@expose("json")
def pagelist(self):
```

- URL per format JSON:

  http://localhost:8080/pagelist?tg_format=json

- Més senzill impossible!

Necessitem ser fashions (AJAX+JavaScript)

- TurboGears utilitza Mochikit per JavaScript
- Afegir al fitxer *gplwiki20/config/app.cfg*

```
tg.include_widgets = ['turbogears.mochikit']
```

  - Reiniciar l'aplicació!

- Afegir els ids on volem tenir AJAX, al *master.kid*

```
<div id="footer">
<p>Veure la <a id="pagelist"
   href="${tg.url('/pagelist')}">completa de
   pàgines.</a></p>
<div id="pagelist_results"></div>
...
```

Necessitem ser fashions (AJAX+JavaScript)

- Script JavaScript per reconstruir la llista de pàgines, al fitxer master.kid:

```
...
</style>
<script type="text/javascript">
addLoadEvent(function(){
    connect($('pagelist'),'onclick', function (e) {
    e.preventDefault();
    var d = loadJSONDoc("${tg.url('/pagelist', tg_format='json')}");
    d.addCallback(showPageList); });});

function showPageList(result) {
    var currentpagelist = UL(null, map(row_display,result["pages"]));
    replaceChildNodes("pagelist_results", currentpagelist);}

function row_display(pagename) {
    return LI(null, A({"href" : "${tg.url('/')}" +pagename},
                                                    pagename))}
</script>
```

Beneficis de TurboGears

- Desenvolupament local
  - No s'ha de pujar l'aplicació a cap servidor
- Temps entre proves mínim!
  - No s'ha de compilar res
  - CherryPy está pendent dels canvis dels fitxers i automàticament recarrega el servidor quan els fitxers font han canviat
- Consultes i actualitzacions a la BD senzilles
  - No es necessari escriure SQL
  - Però es poden escriure, si és necessari
- Els templates són visualitzables al navegador
  - Evita la programació a cegues
- Ahh!! I els widgets! http://localhost:7654/widgets/

Exemples d'aplicacions TurboGears

- Pyndorama - http://code.google.com/p/pyndorama/
- WhatWhat Status - http://code.google.com/p/whatwhat-status/
- VotingPlace.net - http://votingplace.net/
- Doggdot - http://doggdot.us/
- Showmedo - http://showmedo.com/
- GeneSilico - http://genesilico.pl/toolkit/
- PairWise Neighbours - http://genomes.urv.cat/pwneigh/

Agraïments

- Comunitat TurboGears
- Presentació basada en:
  - Documentació TurboGears
  - Documentació d'Stefano Zacchiroli i Christopher Arndt
- GPLTarragona

Moltes gràcies per la vostra atenció!!