

WEB DEVELOPMENT SHOULD BE



Massimo Di Pierro

School of Computing and Digital Media



Chicago, IL

FIRST OF ALL....



Congratulations to Mariano
new member of the PSF

SHOULD EVERYBODY
LEARN TO PROGRAM?

SHOULD EVERY CHILD
LEARN TO PROGRAM?

Yes!

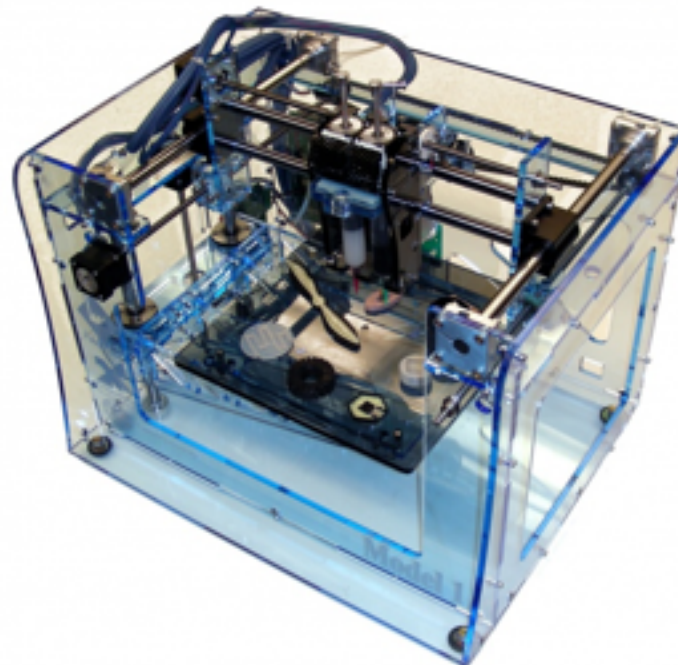
WHY?

- ▶ It improves **cognitive skills**
- ▶ In the next **10 years** software development jobs are projected to increase **2x** as other jobs (in average)
- ▶ We want kids to **understand** technology, not be simply consumers of technology
- ▶ Understanding science and technology is essential for the **progress of society**
- ▶ Computing technologies have given us unprecedented means to **communicate** and build **social relations** but can also be used to take away some of **our rights** (who owns your digital content?)

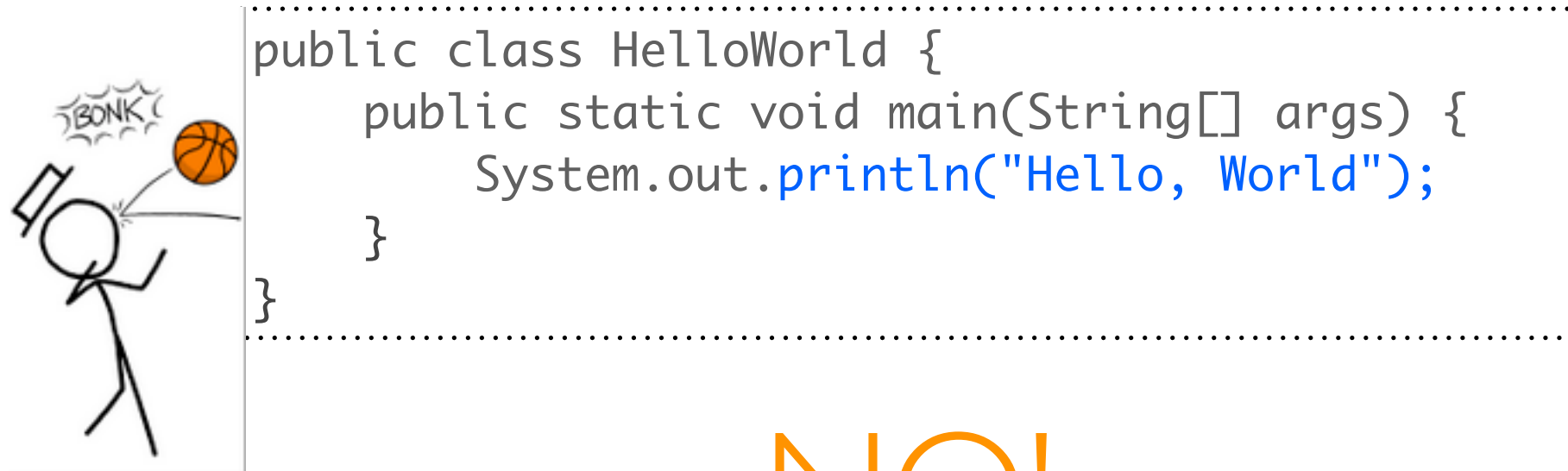
ANYTHING ELSE?

- ▶ It **breaks the old capitalistic model** which makes a distinction between the investors who buy the means of production, and the workers who build products
- ▶ **Anyone can build software** with modest “means of a production”: a laptop, a good education, and time
- ▶ All software developers have the means to build their own companies.
- ▶ They make a product that **has economic value**, often **social value**, can be replicated at no cost, and can be discarded without polluting the environment
- ▶ Production of digital content (especially software) will continue to be an increasing part of the world’ economy.

WHICH SOCIETY DO WE WANT?



DO WE TEACH PROGRAMMING WELL?

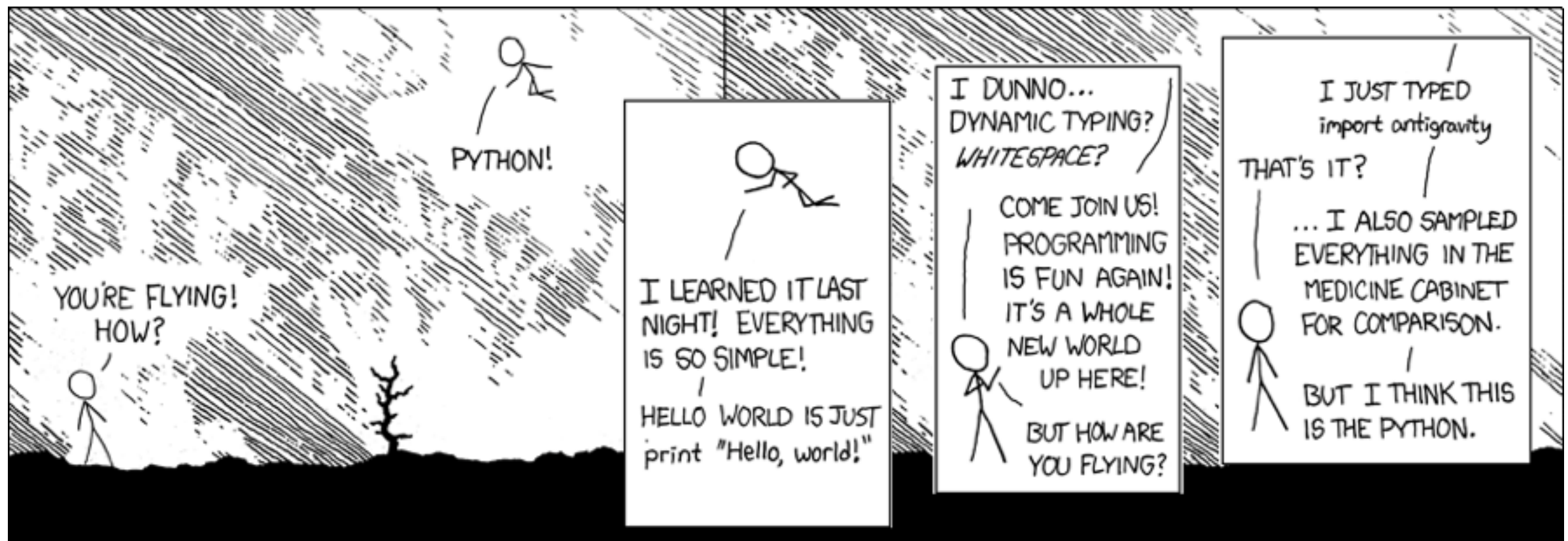


NO!

- ▶ We put obstacles in the way of young programmer (shell, IDE, ...)
- ▶ We use languages that focus on syntax not semantic
- ▶ We use examples that do not leverage on students' knowledge
- ▶ We do not provide motivation (intro classes far from real life)
- ▶ Bottom-up approach instead of top-down approach

HOW CAN WE DO IT BETTER?

```
print "hello world"
```



HOW CAN WE DO IT EVEN BETTER?

```
def index():  
    return "hello world"
```

Put in context: [www!](#)

- ▶ The web provides motivation for learning to program
- ▶ The web gives kids means to communicate
- ▶ Kids already associate the computer with the web

HOW CAN WE DO IT EVEN BETTER?

```
def index():  
    return "hello world"
```

Put in context: www!

- ▶ The web provides motivation for learning to program
- ▶ The web gives kids means to communicate
- ▶ Kids already associate the computer with the web

- ▶ Who here has a kid who is not addicted to youtube?

my job is to make web development **easy**
more accessible
less error prone
more secure
modular

my job is to make web development **easy**
more accessible
less error prone
more secure
modular

Disclaimer: I do not claim any success. I am just trying....

WE2PY: BATTERIES INCLUDED

web server

ssl enabled

DAL + database

auto-migrations

SQLite

web IDE

design, deploy, manage



html, xml, json, rss, ics, pdf, rtf,
xmlrpc, jsonrpc, soap,
ldap, pam, janrain, dropbox, google,
CAS, OpenID, oauth 1&2, x509
marmin, markdown,
google wallet, authorize.net, stripe.com
memcache, redis
twitter bootstrap



.zip

No installation. No configuration. Just Unzip and Click!

WEB2PY CONTRIBUTORS

2011



2012



Web based IDE “admin” with hot plug and play of multiple apps

The screenshot displays the Web2Py admin interface in a web browser. The browser's address bar shows the URL `127.0.0.1:8000/admin/default/site`. The interface features a top navigation bar with a home icon, the text "WEB2PY", and buttons for "Site", "Logout", "Debug", and "Help".

The main content area is titled "INSTALLED APPLICATIONS". It lists three applications:

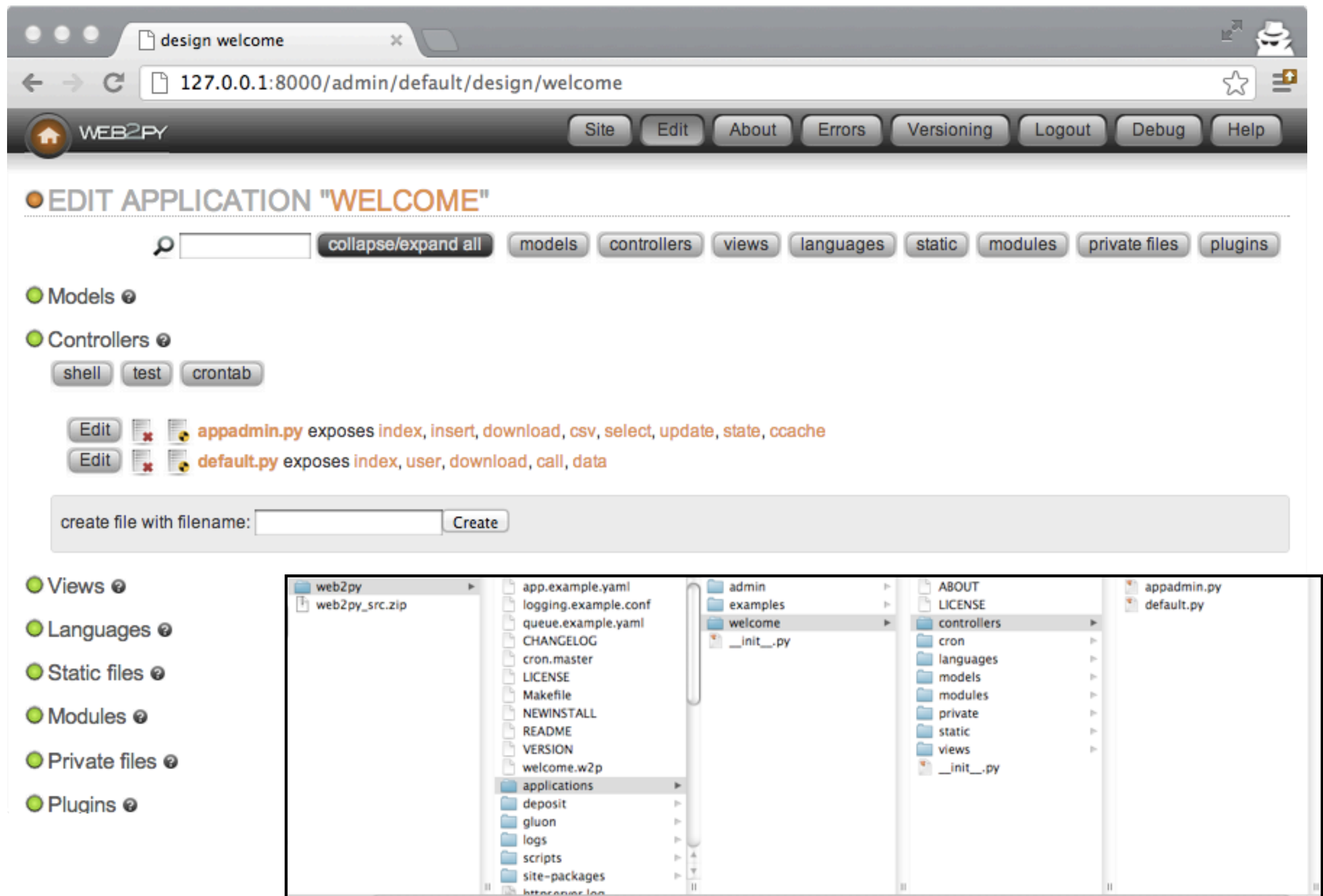
- admin (currently running)**: Includes buttons for "Errors", "Clean", "Pack all", and "Compile".
- examples**: Includes buttons for "Edit", "About", "Errors", and "Clean".
- welcome**: Includes buttons for "Edit", "About", "Errors", and "Clean".

A hand is holding a smartphone in the center, displaying a mobile version of the interface. The phone screen shows a "web2py mobil..." header with a "Home" link, a list of "Installed applications" (admin, examples, welcome), and a footer that says "powered by web2py - ©2012".

On the right side of the interface, there are several sections:

- A "Change admin password" button.
- A status section showing "Version 2.2.1 (2012-11-13 10:24:37) stable" and "Unable to check for upgrades". It also mentions "Running on Rocket 1.2.5" and provides a link to "Try the mobile interface".
- A "New application wizard" section with a "Start wizard" button (noting it "requires internet access").
- A "New simple application" section with an "Application name:" input field and a "Create" button.
- An "Upload and install packed application" section with an "Application name:" input field, an "Upload a package:" section with a "Choose File" button (showing "No file chosen"), and an "Or Get from" input field.

Thin-IDE: only shows file system, no metadata



Web based editor (code-mirror)

The screenshot shows a web browser window with the address bar displaying `127.0.0.1:8000/admin/default/edit/welcome/controllers/default.py?id=controllers__default__py`. The page title is "EDITING FILE 'WELCOME/CONTROLLERS/DEFAULT.PY'". Below the title, there is a summary of the file's actions: "exposes: index, user, download, call, data" and "edit views: index, user". A status bar shows the "Saved file hash: 355e75b88ee524c96b" and "Last saved on: Fri Oct 19 14:12:09 201". To the right of the status bar are buttons for "toggle breakpoint", "<<back", and "docs". The main area displays a Python code editor with line numbers 1 through 19. The code includes a docstring and a function definition. A callout bubble highlights the function definition: `def index():` followed by `return "hello world"` on the next line. The rest of the code in the editor includes a docstring for the `index` function and a call to `auth.wiki()`.

```
1 # -*- coding: utf-8 -*-
2 # this file is released under public domain and you can use it as you wish
3
4 #####
5 ## This is a samples controller
6 ## - index is the default action
7 ## - user is required for authentication
8 ## - download is for downloading the source code
9 ## - call exposes all registered services
10 #####
11
12
13 def index():
14     """
15     example action using the internationalization operator T and flash
16     rendered by views/default/index.html or views/generic.html
17
18     if you need a simple wiki simple replace the two lines below with:
19     return auth.wiki()
```

Web based database administration (per app)

SQLite, MySQL, PostgreSQL, MSSQL, Firebird, Oracle, DB2, Ingres, Informix, Ingres, Sybase, GAE, ...

Twitter Bootstrap Layout

web2py™ Login

- Register
- Lost password?
- Login

Newapp Database Administration (appadmin)

Available Databases and Tables

db.auth_user	New Record
db.auth_group	New Record
db.auth_membership	New Record
db.auth_permission	New Record
db.auth_event	New Record
db.auth_cas	New Record
db.scheduler_task	New Record
db.scheduler_run	New Record
db.scheduler_worker	New Record

Auth Actions

Role Based Access Control Tables

Scheduler Tables

Share

Web translation page for internationalization (per app)

The screenshot shows a web browser window with the address bar displaying `127.0.0.1:8000/admin/default/edit_language/welcome/languages/it.py`. The page title is "EDITING LANGUAGE FILE 'WELCOME/LANGUAGES/IT.PY'". Below the title, there is a navigation bar with buttons: Site, Edit, About, Errors, Versioning, Logout, Debug, and Help. The main content area is titled "ORIGINAL/TRANSLATION" and contains several input fields for editing the language file. Each field has a "delete" button next to it.

Hide/Show Translated strings

ORIGINAL/TRANSLATION

!=

!=

delete

!!langcode!

it

delete

!!langname!

Italiano

delete

"update" is an optional expression like "field1='newvalue'". You cannot update or delete the results of a JOIN

"update" è un'espressione opzionale come "campo1='nuovo valore'". Non si può fare "update" o "delete" dei risultati di un JOIN

delete

%(nrows)s records found

%(nrows)s records found

delete

%d seconds ago

%d seconds ago

delete

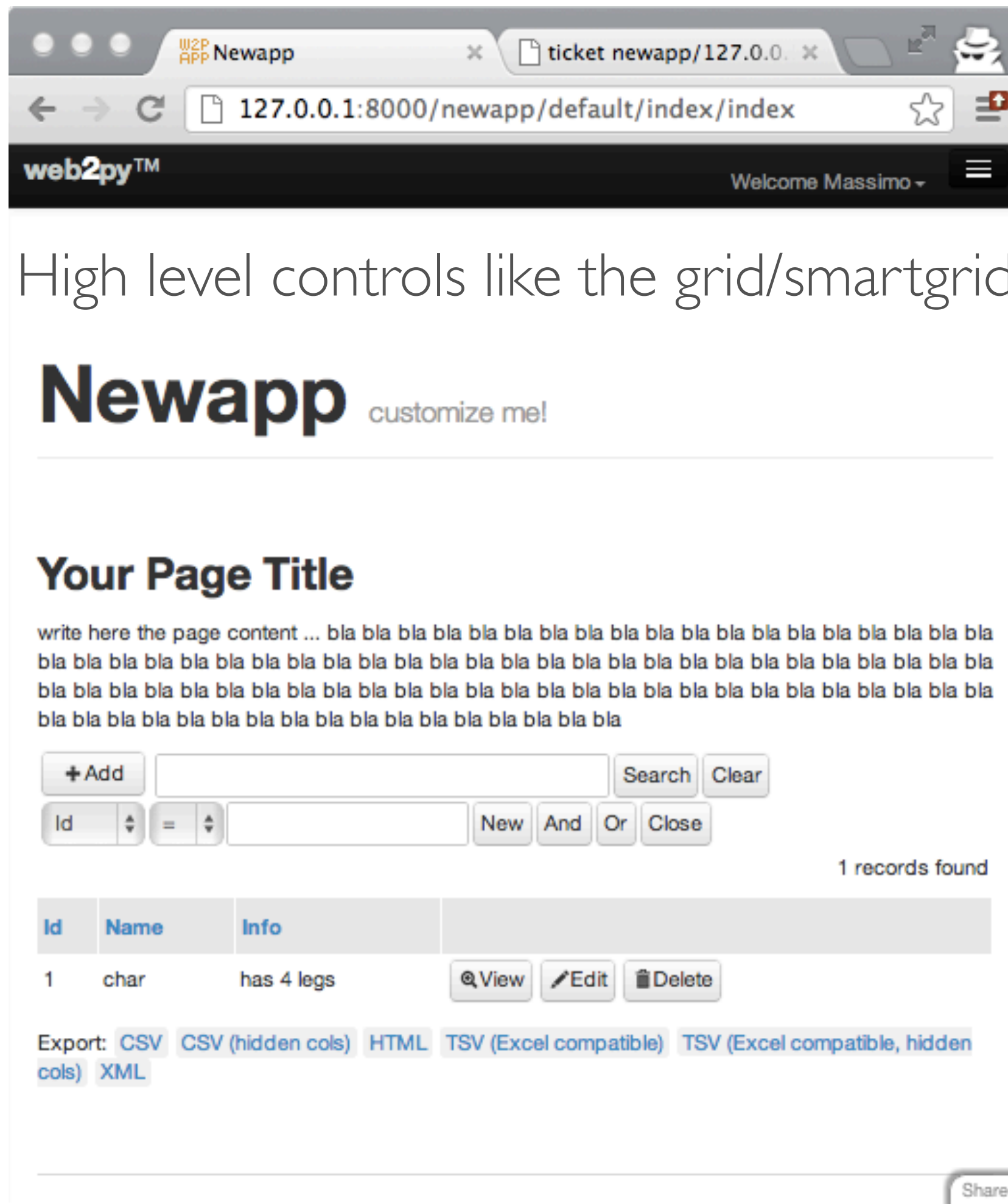
%s %%%{row} deleted

Built-in pluralization system

The screenshot shows a web browser window with the address bar displaying `127.0.0.1:8000/admin/default/edit_plurals/welcome/languages/plural-en.py?nplurals=2`. The page title is "EDITING PLURAL FORMS FILE 'WELCOME/LANGUAGES/PLURAL-EN.PY'". The interface includes a navigation bar with buttons for Site, Edit, About, Errors, Versioning, Logout, Debug, and Help. The main content area displays a table for editing plural forms. The table has three columns: Singular Form, Plural Form #1, and a delete button. The rows contain the following data:

Singular Form	Plural Form #1	
account	accounts	delete
book	books	delete
is	are	delete
man	men	delete
miss	misses	delete
person	people	delete
quark	quarks	delete
shop	shops	delete
this	these	delete
was	were	delete
woman	women	delete

Below the table is an "update" button. At the bottom of the page, a footer bar states: "Powered by web2py™ created by Massimo DI Pierro ©2007-2012 - Admin language English (US)".



SYNTAX



```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello, World");  
    }  
}
```

VS

```
print "hello world"
```

BOTTLE EXAMPLE

```
from bottle import run, route, get, static_file
```

required inputs

```
@get('/index')
```

routing logic

```
def index()
```

action

```
    return 'hello world'
```

```
@route('/static/<filename>')
```

```
def server_static(filename):
```

```
    return static_file(filename, root='static')
```

handler for static files

```
run(host='localhost', port=8080)
```

start web server

FLASK EXAMPLE

```
from flask import Flask, request
```

required inputs

```
app = Flask(__name__)  
app.config.from_object(__name__)
```

boilerplate config
logic

```
@app.route('/index', methods=['GET'])  
def index()  
    return 'hello world'
```

routing logic

action

```
app.run(port=8080)
```

start web server

TORNADO EXAMPLE

```
import tornado.ioloop
import tornado.web
```

required inputs

```
def index(request):
    return 'hello world'
```

action

```
class MainHandler(tornado.web.RequestHandler):
    def get(self): return index(self.request)
```

routing logic

```
application = tornado.web.Application([
    (r"/index", MainHandler),
    (r"/static/(.*)", tornado.web.StaticFileHandler, {"path": "static"})])
```

handler for static files

```
application.listen(8080)
tornado.ioloop.IOLoop.instance().start()
```

start web server

PYRAMID EXAMPLE

```
from wsgiref.simple_server import make_server
from pyramid.config import Configurator
from pyramid.response import Response
from pyramid.static import static_view
```

```
def index(context, request):
    return Response('hello world')
```

```
config = Configurator()
config.add_route('index', '/index')
config.add_view(index, route_name='index')
config.add_static_view(name='static', path='static')
app = config.make_wsgi_app()
server = make_server('0.0.0.0', 8080, app)
server.serve_forever()
```

required inputs

action

routing logic

handler for static
files

start web server

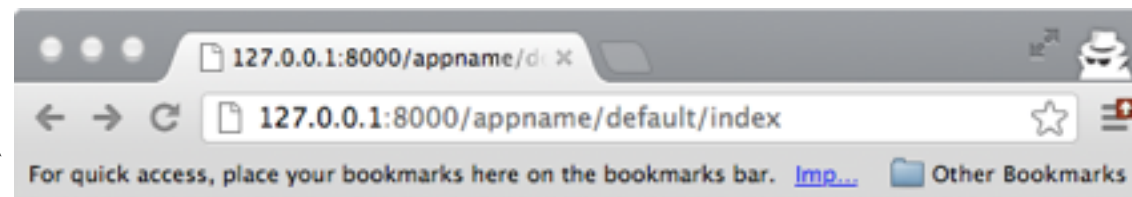
WEB2PY EXAMPLE

<http://127.0.0.1:8000/appname/default/index>

```
def index():  
    return 'hello world'
```

call

action



Hello world

...HUH?



IMPORT VS EXEC



IMPORT VS EXEC

user app

imports



framework

```
from bottle import ...  
from flask import ...  
from tornado import ...  
from pyramid import ...
```

explicit better
than implicit

framework

executes



user app

... app

... app

```
env = build_environment(request)  
app = find_application(request)  
exec app in env (oversimplification)
```

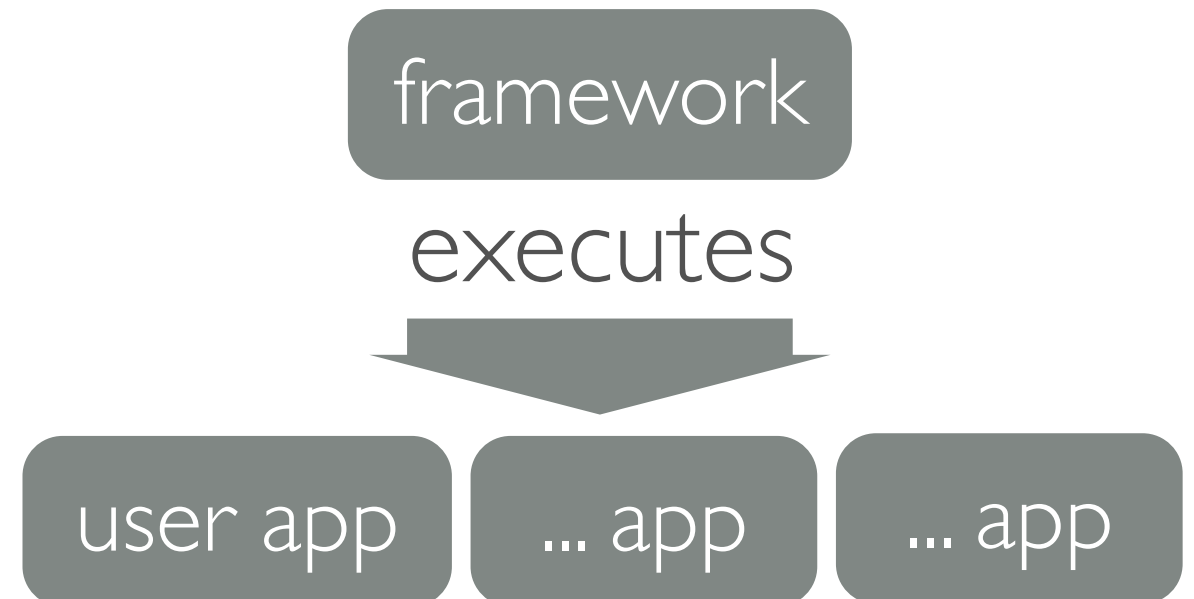
do not repeat
yourself

convention over
configuration

IMPORT VS EXEC



faster (for simple apps)
more flexibility
no “magic”



less code (for simple apps)
how swap of code
multi app/multi project
homogeneous environment
“magic”

LAYERS OF CODE

SQL inside Python
(DAL or ORM)

```
execute('select * from users where id=1')  
db(db.users.id==1).select()
```

HTML inside Python
(helpers)

```
return '<div><h1>%s</h1></div>' % x  
return DIV(H1(x))
```

CODE in HTML
(MVC)

```
<div>{{if x}}check{{endif}}</div>  
<div>{{if x:}}check{{pass}}</div>
```

JS in HTML

```
<div><script>alert('hi!')</script></div>  
<div>{{=LOAD('action',ajax=True)}}</div>
```

WEB2PY DAL

- ▶ SQLite, MySQL, PostgreSQL, MSSQL, Firebird, Oracle, DB2, Ingres, Informix, Ingres, Sybase, GAE, ...
- ▶ automatic migrations
- ▶ multiple dbs, connection pooling, Round Robin redundancy, distributed transactions
- ▶ joins, left joins, aggregates, nested selects, recursive selects

```
db = DAL('sqlite://storage.sqlite')
db.define_table('person', Field('name'))
db.define_table('thing', Field('name'), Field('owner', db.person))
db.thing.insert(name='mac', owner=db.person.insert(name='Max'))

ownership = db.person.id == db.thing.owner
n_things = db.thing.id.count()
for row in db(ownership).select(db.person.name, n_things,
                                groupby=db.person.id):
    print row.person.name, row(n_things)
```

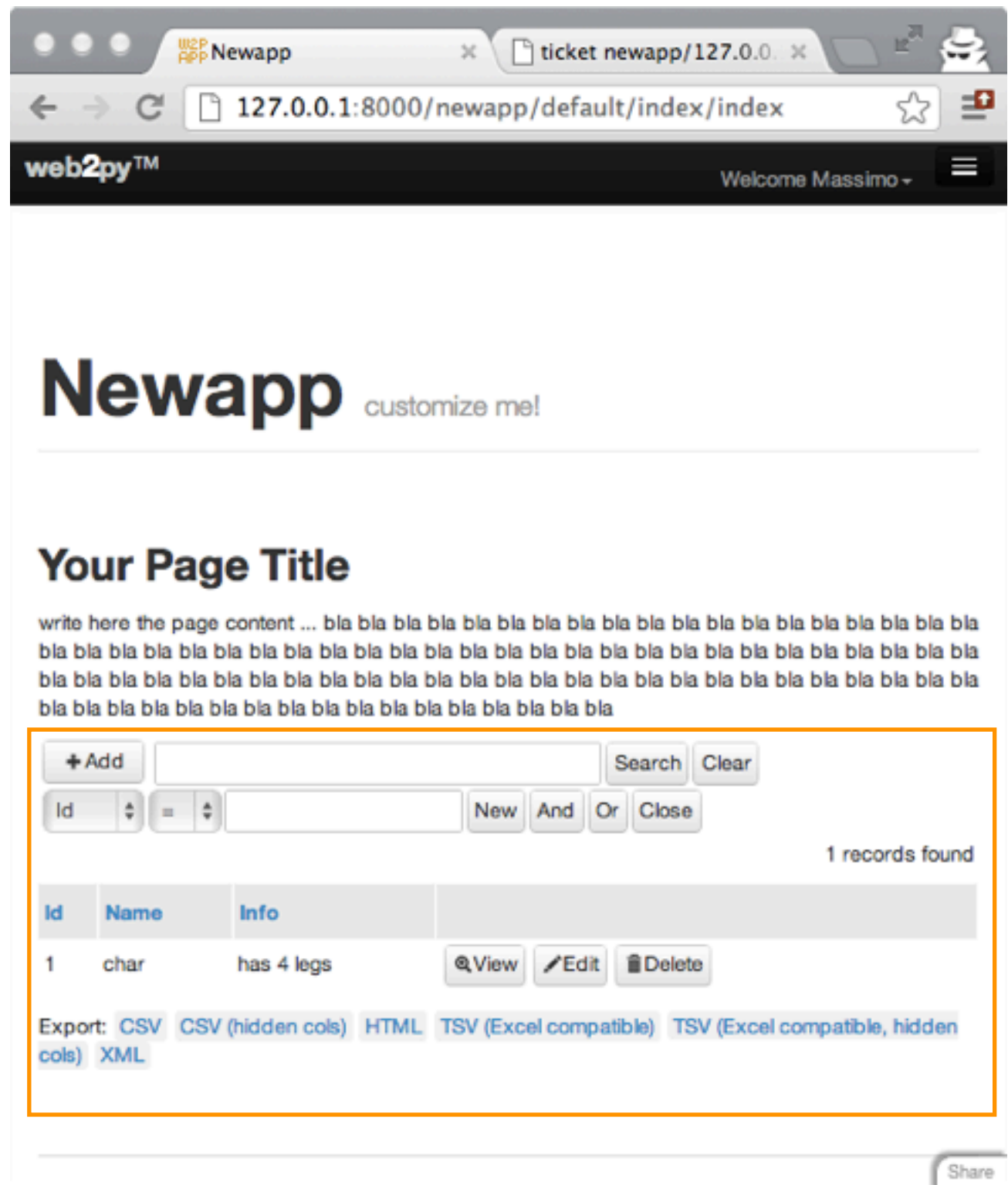
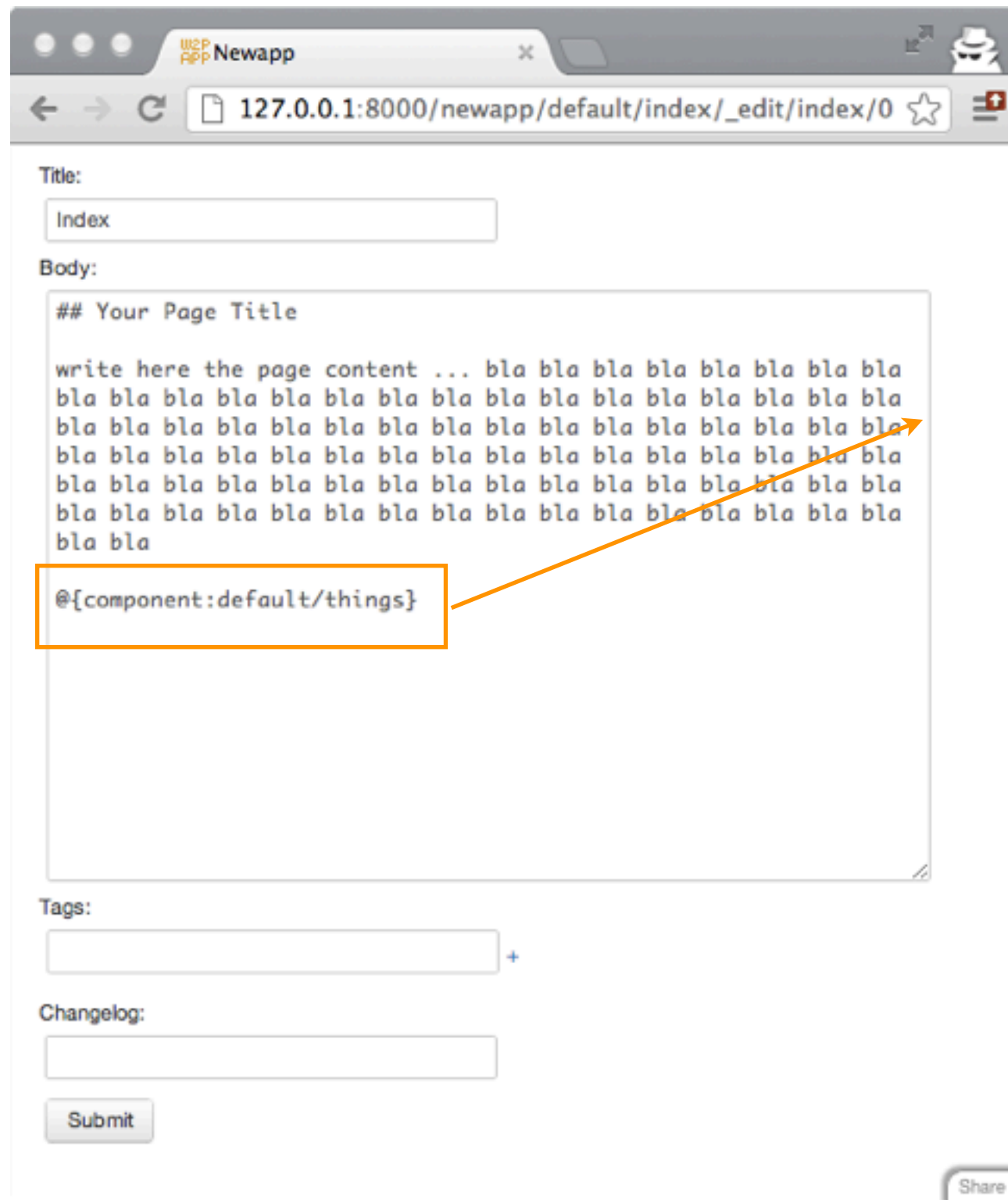
PROGRAMMING AS WIKI

```
# models/db.py
db.define_table('thing',
    Field('name'),
    Field('info', 'test'))

# controllers/default.py
def index():
    return auth.wiki()

def things():
    return SQLFORM.grid(db.thing)
```

PROGRAMMING AS WIKI



CONCLUSIONS

- ▶ The elitist approach to programming leads us to the wrong path
- ▶ There is not only one solution
- ▶ There is not only one web framework
- ▶ We need to learn from each other
- ▶ We need to build a society where technology is controller by people, not by large corporation
- ▶ We need to build tools that are easier to user and effective and allow more people to use technology for public good