Modern Web Application Framework

Python, SQL Alchemy, Jinja2 & Flask

Devert Alexandre

Table of Contents

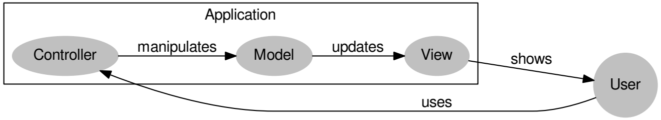
Model-View-Controller

Most of the modern web development frameworks follow the *Model-View-Controller* model (MVC model)

- The *model* : representation of data. Usually, have a strong relation with the database
- The *views* : what is shown to the user. Can be any kind of user interface, usually HTML pages with Javascript.
- The *controls* : what operation are done on the data.

It's a rather convenient way to design software projects involving user interfaces presenting and manipulating data.

Model-View-Controller



Model-View-Controller

Example for *Model-View-Controller* : an online management game

- The rule of the game, updating the state of each player $\Rightarrow$ the model
- The HTML pages, showing the various screen of the game $\Rightarrow$ the views
- The methods called when a user click on the screen $\Rightarrow$ the controllers

Model-View-Controller

Example for *Model-View-Controller* : an online shop

- The list of products, the payment rules, delivery orders $\Rightarrow$ the model
- The HTML pages, showing the various screen of the shop $\Rightarrow$ the views
- The methods for payment, order, shopping cart $\Rightarrow$ the controllers

Model-View-Controller

Model-View-Controller also helps to organize the work

- Some work on the views $\Rightarrow$ graphic designers, HTML, javascript
- Some work on the model $\Rightarrow$ database, software architecture
- Some work on the controls $\Rightarrow$ rather low-level and/or specialized code
- Some work on writing unit tests for at least the model and the views

Table of Contents

Web application with script language

Why using a scripting language for a web application ?

- More adapted language to paste together various components (database, rendering, routing, . . .)
- Make its easier to release early & often
- Easier to maintain & modify
- Speed far enough for many use case

Web application with script language

Why not PHP, or PHP framework ?

- Designed to make simple web pages, not large web applications
- *Awfully* designed programming language
- very inconsistent libraries
- very little help for debugging
- many security issues
- many better alternatives

Detailed explanation here *http://me.veekun.com/blog/2012/04/09/php-a-fractal-of-bad-design*

Web application with script language

Why not using Java/JSP/JBoss/Apache/Hibernate/Spring ?

- Even simple changes requires lots of coding
- Big changes takes a lot of planning
- Edit/Compile/Run takes more ressource
- General speed of development much reduced
- Working without a big fat IDE is tedious

But you can use those all this with a script-like language : *Grails* and *Groovy*

Flask

I am going to introduce the framework *Flask*

- It is small : quick to learn and master
- It is complete : you can use to do serious apps
- It is lean : a shell and a text editor are enough, no need for an IDE to be efficient with it
- It is very well documented

The same ideas can be found in most web development frameworks.

Flask

Flask is a nice glue around existing tools

- *Python* $\Rightarrow$ programming language
- *SQL Alchemy* $\Rightarrow$ database
- *Jinja2* $\Rightarrow$ HTML template system
- *Werkzeug* $\Rightarrow$ WSCGI handling (CGI, but better)

Table of Contents

Hello, world !

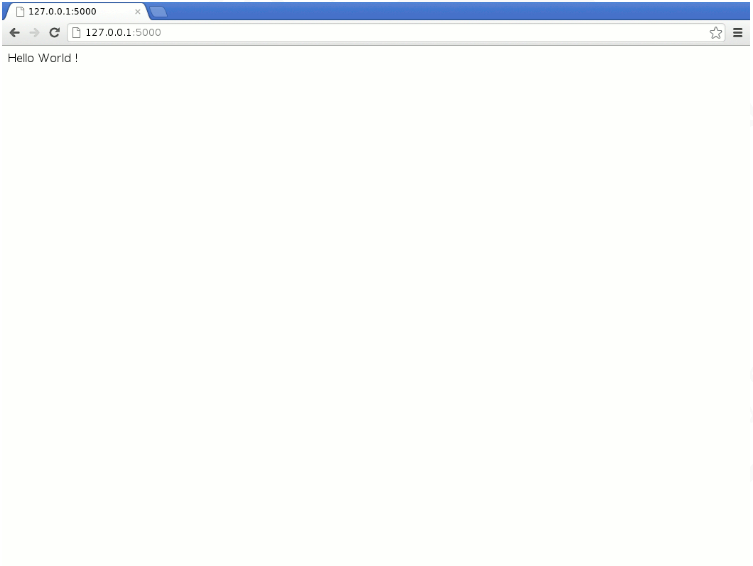
A minimal Flask application

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello():
  return 'Hello World !'

if __name__ == '__main__':
  app.run()
```

Run this, and open your web browser at *http://127.0.0.1:5000*

Hello, world !

You will see this



Hello World !

Hello, world !

This creates an *application* instance and run it

```
from flask import Flask
app = Flask(__name__)

if __name__ == '__main__':
  app.run()
```

Hello, world !

This adds the *hello* method to the application instance

```
@app.route('/')
def hello():
  return 'Hello World !'
```

- *hello()* will be called every time the address / is requested
- *hello()* returns the text data for the web browser

Debugging

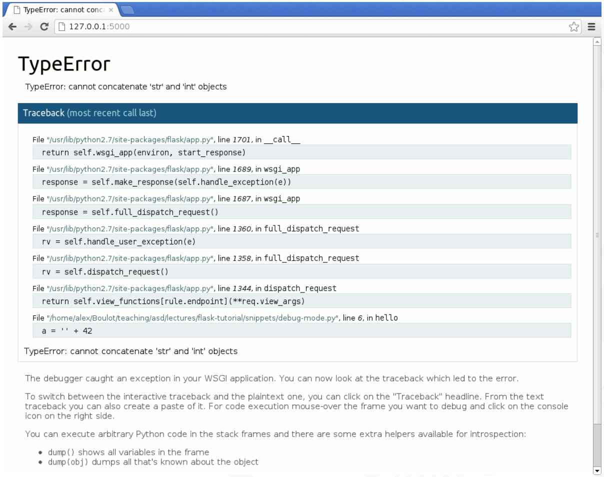
Triggering the debug mode is easy

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello():
  return 'Hello World !'

if __name__ == '__main__':
  app.run(debug = True)
```

In debug mode, you can edit the code while the server runs : it will restart !

Debugging

The debug mode will also helps a lot to point where the problem is



TypeError

TypeError: cannot concatenate 'str' and 'int' objects

Traceback (most recent call last)

```
File "/usr/lib/python2.7/site-packages/flask/app.py", line 1701, in __call__
  return self.wsgi_app(environ, start_response)
File "/usr/lib/python2.7/site-packages/flask/app.py", line 1689, in wsgi_app
  response = self.make_response(self.handle_exception(e))
File "/usr/lib/python2.7/site-packages/flask/app.py", line 1687, in wsgi_app
  response = self.full_dispatch_request()
File "/usr/lib/python2.7/site-packages/flask/app.py", line 1360, in full_dispatch_request
  rv = self.handle_user_exception(e)
File "/usr/lib/python2.7/site-packages/flask/app.py", line 1358, in full_dispatch_request
  rv = self.dispatch_request()
File "/usr/lib/python2.7/site-packages/flask/app.py", line 1344, in dispatch_request
  return self.view_functions[rule.endpoint](**req.view_args)
File "/home/alex/Boulot/teaching/asd/lectures/flask-tutorial/snippets/debug-mode.py", line 6, in hello
  a = '' + 42
TypeError: cannot concatenate 'str' and 'int' objects
```

The debugger caught an exception in your WSGI application. You can now look at the traceback which led to the error.

To switch between the interactive traceback and the plaintext one, you can click on the "Traceback" headline. From the text traceback you can also create a paste of it. For code execution mouse-over the frame you want to debug and click on the console icon on the right side.

You can execute arbitrary Python code in the stack frames and there are some extra helpers available for introspection:

- `dump()` shows all variables in the frame
- `dump(obj)` dumps all that's known about the object

Table of Contents

Function / URL mapping

When an URL is requested, Flask will look for its corresponding function.

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def index():
    return 'Index Page'

@app.route('/welcome')
def hello():
    return 'Hello World'

if __name__ == '__main__':
  app.run()
```

One function return text data. It can be HTM, XML, JSON, etc.

Function / URL mapping

You can defines URL with parameters

```
@app.route('/show_name/<name>')
def print_name(name):
  return 'Hello, %s !' % name
```

It gives a nice way, intuitive way to define ressources on a website.

Function / URL mapping

You can make URL parameters optional

```
@app.route('/hello/')
@app.route('/hello/<name>')
def hello(name = None):
  if name is None:
    return 'A horse with no name'
  else:
    return 'A horse named %s' % name
```

Function / URL mapping

You can enforce the type of a parameter

```
@app.route('/team/<int:team_id>')
def show_team(team_id):
  return 'team #%d' % team_id
```

Flask will check the type for you

Function / URL mapping

You can translate function names to URL with *url_for()*

```
@app.route('/')
def welcome():
  return 'Hello World !'

@app.route('/test')
def test():
  name = 'welcome'
  return 'url_for "%s" = "%s"' % (name, url_for(name))
```

Especially convenient when you might have to change the URL naming scheme

Function / URL mapping

url_for() also works for URL with parameters

```
@app.route('/show_name/<name>')
def print_name(name):
  return 'Hello, %s !' % name

@app.route('/test')
def test():
  func_name, user_name = 'print_name', 'Alex'
  return 'url_for "%s" = "%s"' % (func_name, url_for(func_name, name = user_name))
```

Catching HTTP errors

The HTTP protocol defines several status codes.

status code	*meaning*
400	Bad Request
401	Unauthorized
402	Payment Required
403	Forbidden
404	Not Found
500	Internal Server Error
501	Not Implemented
503	Service Unavailable

Catching HTTP errors

Using *@errorhandler*, you can catch such errors

```
@app.errorhandler(403)
def page_forbidden(error):
  print 'Hey ! You are not allowed to access this !'

@app.errorhandler(404)
def page_not_found(error):
  print 'Ho no ! The ressource you want to access does not exist :('
```

Throwing HTTP errors

It is also possible to throw HTTP errors with *abort*

```
@app.route('/show_account_infos')
def show_account_infos():
  if not user.logged_in:
    abort(401)

  # Do things ...
```

For instance, an error 401 to deny access to ressources

Table of Contents

The need for templates

Generating HTML directly with code

- Easy to make very hard to read code
- Mix-up the *control* code with the *view* code

Text template system is a convenient and common way to separade the *view* code from the remaining code

The need for templates

Flask uses Jinja2 as template system. There are many others template system

- Mako, for Python (if you ask me, it's better than Jinja2)
- JSP, for Java, THE standard for Java. Allow to mix Java & HTML.
- ASP, for Microsoft products. Allow to mix VBScript & HTML.
- XSLT is a template system based on XML. Plateform indepedent but not very convenient in practice.
- Maybe 10 different for every language you can think of

Basic template rendering

The function *render_template* takes a path to an HTML file, and arbitrary parameters

```
from flask import Flask, render_template
app = Flask(__name__)

@app.route('/hello/')
@app.route('/hello/<name>')
def hello(name = None):
  return render_template('hello.html', name = name)

if __name__ == '__main__':
  app.run()
```

What will be returned will the content of *hello.html*

Basic template rendering

The HTML file *hello.html*

```
<!doctype html>
<html>
  <head>
    <title>The website that says Hello to you</title>
  </head>
  <body>
    {% if name %}
    <h1>Hello , {{ name }} !</h1>
    {% else %}
    <h1>Hello , thing with no name !</h1>
    {% endif %}
  </body>
</html>
```

It's no ordinary HTML ⇒ there are instruction mixed in !

Basic template rendering

The HTML file *hello.html*

```
<!doctype html>
<html>
  <head>
    <title>The website that says Hello to you</title>
  </head>
  <body>
    {% if name %}
    <h1>Hello, {{ name }} !</h1>
    {% else %}
    <h1>Hello, thing with no name !</h1>
    {% endif %}
  </body>
</html>
```

hello.html is processed to generate the HTML to send to a user. Here, we use the *name* variable, passed as a parameter of *render_template*

Basic template rendering

The HTML file *hello.html*

```
<!doctype html>
<html>
  <head>
    <title>The website that says Hello to you</title>
  </head>
  <body>
    {% if name %}
    <h1>Hello , {{ name }} !</h1>
    {% else %}
    <h1>Hello , thing with no name !</h1>
    {% endif %}
  </body>
</html>
```

Variables values can be rendered to text with {{ }}

Basic template rendering

The HTML file *hello.html*

```
<!doctype html>
<html>
  <head>
    <title>The website that says Hello to you</title>
  </head>
  <body>
    {% if name %}
    <h1>Hello , {{ name }} !</h1>
    {% else %}
    <h1>Hello , thing with no name !</h1>
    {% endif %}
  </body>
</html>
```

Blocks of code are put between {% %}

Basic template rendering

Flask assumes that all your templates will be in a *template* directory, relative to your script

```
|— templates
|    |
|    |— hello.html
|
|— test.py
```

Using ressources

If you wish to use other file ressources, like pictures or CSS files, you can put them in directory named *static*

```
|— templates
|    |
|    |— hello.html
|
|— static
|    |
|    |— style.css
|
|— test.py
```

Those resource are not dynamic, not generated on the fly like the HTML code, hence the name "static"

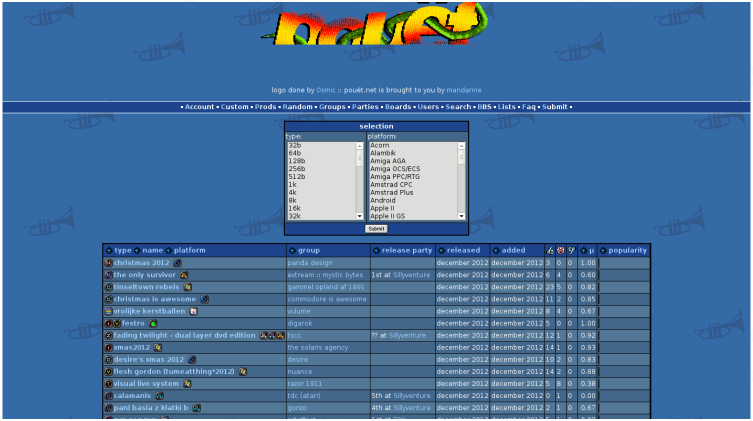
Using ressources

Then, to use those ressources, you can again use *url_for*

```
<!doctype html>
<html>
  <head>
    <title>The website that says Hello to you</title>
    <link rel=stylesheet type=text/css
          href="{{ url_for('static', filename='style.css') }}">
  </head>
  <body>
    {% if name %}
    <h1>Hello, {{ name }} !</h1>
    {% else %}
    <h1>Hello, thing with no name !</h1>
    {% endif %}
  </body>
</html>
```

Template inheritance

On a typical website, different views follow a similar design

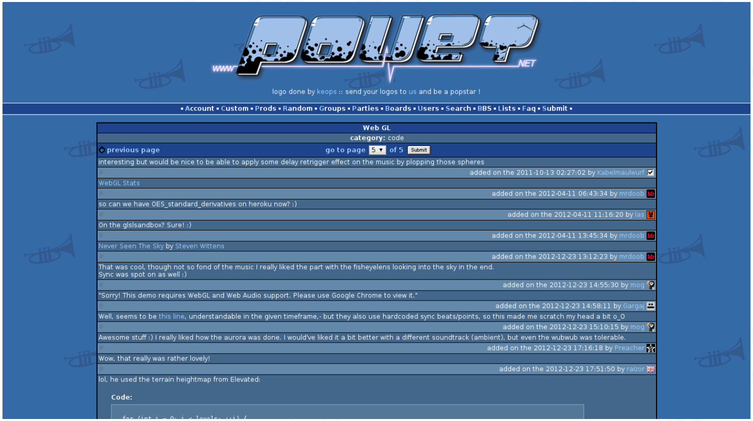


logo done by Osmic :: pouët.net is brought to you by mandarine

type name platform	group	release party	released	added	👍	🐷	👎	µ	popularity
christmas 2012	panda design		december 2012	december 2012	3	0	0	1.00	
the only survivor	extream :: mystic bytes	1st at Sillyventure	december 2012	december 2012	6	4	0	0.60	
tinseltown rebels	gammel opland af 1891		december 2012	december 2012	23	5	0	0.82	
christmas is awesome	commodore is awesome		december 2012	december 2012	11	2	0	0.85	
vrolijke kerstballen	vulume		december 2012	december 2012	8	4	0	0.67	
festro	digarok		december 2012	december 2012	5	0	0	1.00	
fading twilight - dual layer dvd edition	tscc	?? at Sillyventure	december 2012	december 2012	12	1	0	0.92	
xmas2012	the solaris agency		december 2012	december 2012	14	1	0	0.93	
desire's xmas 2012	desire		december 2012	december 2012	10	2	0	0.83	
flesh gordon (tumeatthing*2012)	nuance		december 2012	december 2012	14	2	0	0.88	
visual live system	razor 1911		december 2012	december 2012	5	8	0	0.38	
calamanis	tdc (atari)	5th at Sillyventure	december 2012	december 2012	0	1	0	0.00	
pani basia z klatki b	gonzo	4th at Sillyventure	december 2012	december 2012	2	1	0	0.67	

Template inheritance

On a typical website, different views follow a similar design



logo done by keops :: send your logos to us and be a popstar !

▪ Account ▪ Custom ▪ Prods ▪ Random ▪ Groups ▪ Parties ▪ Boards ▪ Users ▪ Search ▪ BBS ▪ Lists ▪ Faq ▪ Submit ▪

Web GL
category: code

previous page   go to page 5 of 5 Submit

interesting but would be nice to be able to apply some delay retrigger effect on the music by plopping those spheres
added on the 2011-10-13 02:27:02 by Kabelmaulwurf

WebGL Stats
added on the 2012-04-11 06:43:34 by mrdoob

so can we have OES_standard_derivatives on heroku now? :)
added on the 2012-04-11 11:16:20 by las

On the glslsandbox? Sure! :)
added on the 2012-04-11 13:45:34 by mrdoob

Never Seen The Sky by Steven Wittens
added on the 2012-12-23 13:12:23 by mrdoob

That was cool, though not so fond of the music I really liked the part with the fisheyelens looking into the sky in the end.
Sync was spot on as well :)
added on the 2012-12-23 14:55:30 by mog

"Sorry! This demo requires WebGL and Web Audio support. Please use Google Chrome to view it."
added on the 2012-12-23 14:58:11 by Gargaj

Well, seems to be this line, understandable in the given timeframe,- but they also use hardcoded sync beats/points, so this made me scratch my head a bit o_O
added on the 2012-12-23 15:10:15 by mog

Awesome stuff :) I really liked how the aurora was done. I would've liked it a bit better with a different soundtrack (ambient), but even the wubwub was tolerable.
added on the 2012-12-23 17:16:18 by Preacher

Wow, that really was rather lovely!
added on the 2012-12-23 17:51:50 by raizor

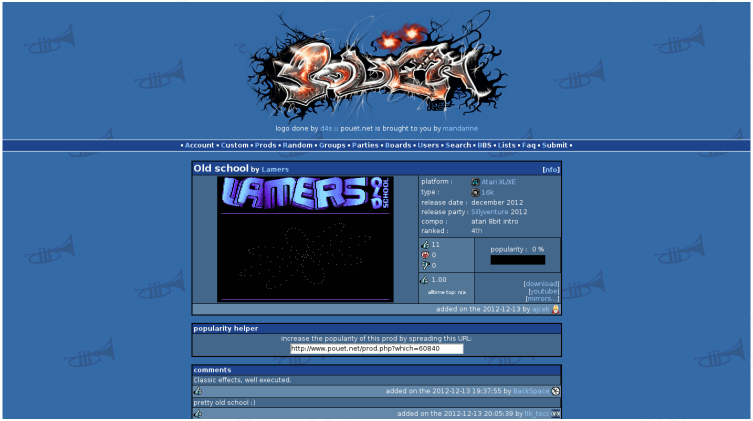
lol, he used the terrain heightmap from Elevated:

Code:

```
for (int i = 0; i < levels; ++i) {
```

Template inheritance

On a typical website, different views follow a similar design



logo done by d4s :: pouët.net is brought to you by mandarine

▪ Account ▪ Custom ▪ Prods ▪ Random ▪ Groups ▪ Parties ▪ Boards ▪ Users ▪ Search ▪ BBS ▪ Lists ▪ Faq ▪ Submit ▪

Old school by Lamers [nfo]

platform :	Atari XL/XE
type :	16k
release date :	december 2012
release party :	Sillyventure 2012
compo :	atari 8bit intro
ranked :	4th

11
0
0
popularity : 0 %
1.00
alltime top: n/a
[download]
[youtube]
[mirrors...]

added on the 2012-12-13 by ajcek

popularity helper

increase the popularity of this prod by spreading this URL:
http://www.pouet.net/prod.php?which=60840

comments

Classic effects, well executed.

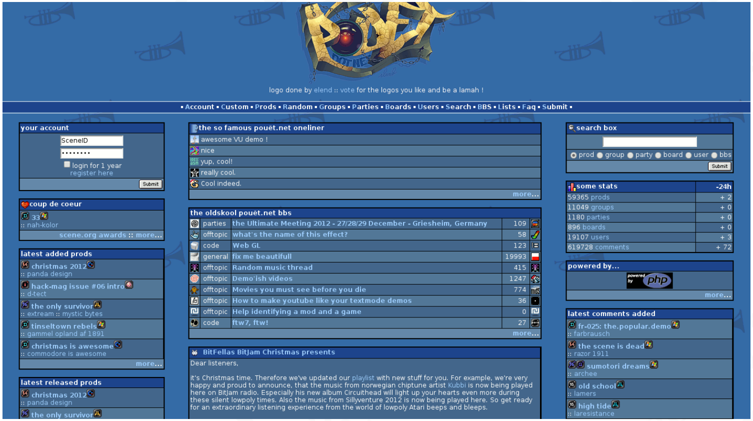
added on the 2012-12-13 19:37:55 by BackSpace

pretty old school :)

added on the 2012-12-13 20:05:39 by ltk_tscc

Template inheritance

On a typical website, different views follow a similar design



logo done by elend :: vote for the logos you like and be a lamah !

▪ Account ▪ Custom ▪ Prods ▪ Random ▪ Groups ▪ Parties ▪ Boards ▪ Users ▪ Search ▪ BBS ▪ Lists ▪ Faq ▪ Submit ▪

your account

SceneID

••••••••

☐ login for 1 year

register here

Submit

coup de coeur

33
:: nah-kolor

scene.org awards :: more...

latest added prods

christmas 2012
:: panda design

hack-mag issue #06 intro
:: d-tect

the only survivor
:: extream :: mystic bytes

tinseltown rebels
:: gammel opland af 1891

christmas is awesome
:: commodore is awesome

more...

latest released prods

christmas 2012
:: panda design

the only survivor

the so famous pouët.net oneliner

awesome VU demo !
nice
yup, cool!
really cool.
Cool indeed.

more...

the oldskool pouët.net bbs

parties	the Ultimate Meeting 2012 - 27/28/29 December - Griesheim, Germany	109
offtopic	what's the name of this effect?	58
code	Web GL	123
general	fix me beautifull	19993
offtopic	Random music thread	415
offtopic	Demo'ish videos	1247
offtopic	Movies you must see before you die	774
offtopic	How to make youtube like your textmode demos	36
offtopic	Help identifying a mod and a game	0
code	ftw7, ftw!	27

more...

BitFellas BitJam Christmas presents

Dear listeners,

it's Christmas time. Therefore we've updated our playlist with new stuff for you. For example, we're very happy and proud to announce, that the music from norwegian chiptune artist Kubbi is now being played here on BitJam radio. Especially his new album Circuithead will light up your hearts even more during these silent lowpoly times. Also the music from Sillyventure 2012 is now being played here. So get ready for an extraordinary listening experience from the world of lowpoly Atari beeps and bleeps.

search box

[✓ prod] group party board user bbs

Submit

some stats	-24h
59365 prods	+ 2
11049 groups	+ 0
1180 parties	+ 0
896 boards	+ 0
19107 users	+ 3
619728 comments	+ 72

powered by...

powered by php

more...

latest comments added

fr-025: the.popular.demo
:: farbrausch

the scene is dead
:: razor 1911

sumotori dreams
:: archee

old school
:: lamers

high tide
:: laresistance

Template inheritance

Jinja2 provides a simple way to share a common template and specialize it : *template inheritance*

```
{% extends "base.html" %}

{% block content %}
    {% if name %}
    <h2>Hello, {{ name }} !</h2>
    {% else %}
    <h2>Hello, thing with no name !</h2>
    {% endif %}
{% endblock %}
```

hello.html extends *base.html*

Template inheritance

Jinja2 provides a simple way to share a common template and specialize it : *template inheritance*

```
{% extends "base.html" %}

{% block content %}
    {% if name %}
    <h2>Goodbye, {{ name }} !</h2>
    {% else %}
    <h2>Goodbye, thing with no name !</h2>
    {% endif %}
{% endblock %}
```

goodbye.html extends *base.html*

Template inheritance

And *base.html* look like this

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html lang="en">
  <head>
    <title>Salute.com, the website that salutes you</title>
    <link rel=stylesheet type=text/css href="{{ url_for('static', filename='style.css
  </head>
  <body>
    <div id="container">
      <div id="header">
        <h1>Salute.com</h1>
        <p>The website that salutes you</p>
      </div>

      <div id="content">
{% block content %}{% endblock %}
      </div>
    </div>

    <div id="footer">
      <h2>Salute.com</h2>
      <p>Site design &amp; copyright &copy; Alexandre Devert</p>
    </div>
  </body>
</html>
```

Template inheritance

On the Python side, *hello.html* and *goodbye.html* are just normal HTML pages, nothing special to do

```
@app.route('/hello/')
@app.route('/hello/<name>')
def hello(name = None):
  return render_template('hello.html', name = name)

@app.route('/goodbye/')
@app.route('/goodbye/<name>')
def goodbye(name = None):
  return render_template('goodbye.html', name = name)
```

Template inheritance

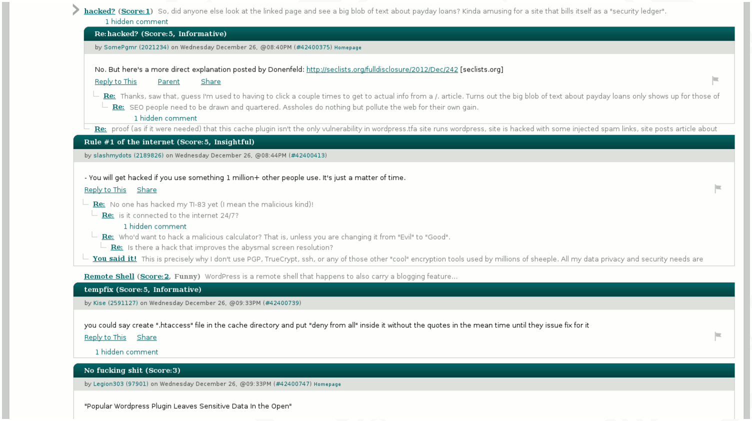
In this exemple, extending *base.html* provides

- A common title
- Includes common ressources (css, javascript, etc.)
- A common header
- A common footer
- The specialized part goes in the "content" block.

Coherent look, code reusage, and clean separation !

Template macros

On a website, the same user interface elements are often re-used



hacked? (**Score:1**) So, did anyone else look at the linked page and see a big blob of text about payday loans? Kinda amusing for a site that bills itself as a "security ledger".
1 hidden comment

Re:hacked? (Score:5, Informative)
by SomePgmr (2021234) on Wednesday December 26, @08:40PM (#42400375) Homepage

No. But here's a more direct explanation posted by Donenfeld: http://seclists.org/fulldisclosure/2012/Dec/242 [seclists.org]
Reply to This Parent Share

Re: Thanks, saw that, guess I'm used to having to click a couple times to get to actual info from a /. article. Turns out the big blob of text about payday loans only shows up for those of
Re: SEO people need to be drawn and quartered. Assholes do nothing but pollute the web for their own gain.
1 hidden comment
Re: proof (as if it were needed) that this cache plugin isn't the only vulnerability in wordpress.tfa site runs wordpress, site is hacked with some injected spam links, site posts article about

Rule #1 of the internet (Score:5, Insightful)
by slashmydots (2189826) on Wednesday December 26, @08:44PM (#42400413)

- You will get hacked if you use something 1 million+ other people use. It's just a matter of time.
Reply to This Share

Re: No one has hacked my TI-83 yet (I mean the malicious kind)!
Re: is it connected to the internet 24/7?
1 hidden comment
Re: Who'd want to hack a malicious calculator? That is, unless you are changing it from "Evil" to "Good".
Re: Is there a hack that improves the abysmal screen resolution?
You said it! This is precisely why I don't use PGP, TrueCrypt, ssh, or any of those other "cool" encryption tools used by millions of sheeple. All my data privacy and security needs are

Remote Shell (**Score:2**, **Funny**) WordPress is a remote shell that happens to also carry a blogging feature...

tempfix (Score:5, Informative)
by Kise (2591127) on Wednesday December 26, @09:33PM (#42400739)

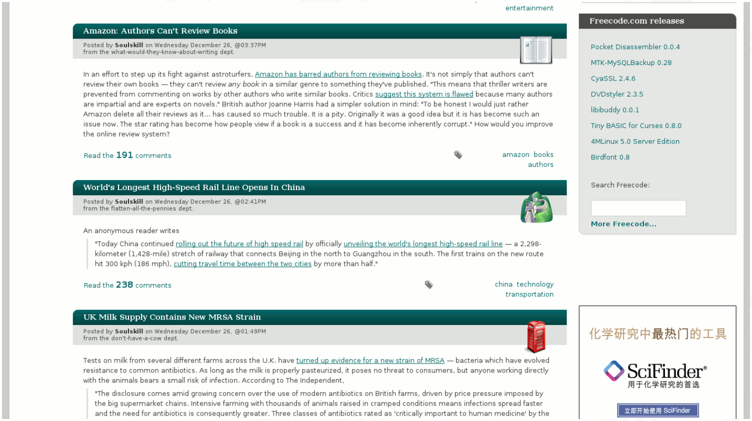
you could say create ".htaccess" file in the cache directory and put "deny from all" inside it without the quotes in the mean time until they issue fix for it
Reply to This Share
1 hidden comment

No fucking shit (Score:3)
by Legion303 (97901) on Wednesday December 26, @09:33PM (#42400747) Homepage

"Popular Wordpress Plugin Leaves Sensitive Data In the Open"

Template macros

On a website, the same user interface elements are often re-used


entertainment

Amazon: Authors Can't Review Books

Posted by **Soulskill** on Wednesday December 26, @03:37PM
from the what-would-they-know-about-writing dept.

In an effort to step up its fight against astroturfers, Amazon has barred authors from reviewing books. It's not simply that authors can't review their own books — they can't review *any book* in a similar genre to something they've published. "This means that thriller writers are prevented from commenting on works by other authors who write similar books. Critics suggest this system is flawed because many authors are impartial and are experts on novels." British author Joanne Harris had a simpler solution in mind: "To be honest I would just rather Amazon delete all their reviews as it… has caused so much trouble. It is a pity. Originally it was a good idea but it is has become such an issue now. The star rating has become how people view if a book is a success and it has become inherently corrupt." How would you improve the online review system?

Read the **191** comments

amazon books authors

World's Longest High-Speed Rail Line Opens In China

Posted by **Soulskill** on Wednesday December 26, @02:41PM
from the flatten-all-the-pennies dept.

An anonymous reader writes

> "Today China continued rolling out the future of high speed rail by officially unveiling the world's longest high-speed rail line — a 2,298-kilometer (1,428-mile) stretch of railway that connects Beijing in the north to Guangzhou in the south. The first trains on the new route hit 300 kph (186 mph), cutting travel time between the two cities by more than half."

Read the **238** comments

china technology transportation

UK Milk Supply Contains New MRSA Strain

Posted by **Soulskill** on Wednesday December 26, @01:49PM
from the don't-have-a-cow dept.

Tests on milk from several different farms across the U.K. have turned up evidence for a new strain of MRSA — bacteria which have evolved resistance to common antibiotics. As long as the milk is properly pasteurized, it poses no threat to consumers, but anyone working directly with the animals bears a small risk of infection. According to The Independent,

> "The disclosure comes amid growing concern over the use of modern antibiotics on British farms, driven by price pressure imposed by the big supermarket chains. Intensive farming with thousands of animals raised in cramped conditions means infections spread faster and the need for antibiotics is consequently greater. Three classes of antibiotics rated as 'critically important to human medicine' by the

Freecode.com releases

- Pocket Disassembler 0.0.4
- MTK-MySQLBackup 0.28
- CyaSSL 2.4.6
- DVDstyler 2.3.5
- libibuddy 0.0.1
- Tiny BASIC for Curses 0.8.0
- 4MLinux 5.0 Server Edition
- Birdfont 0.8

Search Freecode:

More Freecode...

化学研究中最热门的工具

SciFinder®
用于化学研究的首选

立即开始使用 SciFinder

Template macros

We can define reusable HTML bits of codes.

```
{%- macro render_panel(title, style="left") %}
<div class="panel">
  <h1 class="{{ style }}">{{ title }}</h1>
  <div class="panel-content">
    <div class="{{ style }}">
      {{ caller() }}
    </div>
  </div>
</div>
{%- endmacro %}
```

This define a box, containing whatever *caller()* will put in it, and with a title. We put this in *ui.html*

Template macros

Now, we can create lots of boxes.

```
{% extends "base.html" %}
{% import "ui.html" as ui %}

{% block content %}
<div class="three-columns-layout">
  <div class="left-column">
    {% call ui.render_panel("Lorem ipsum", "left") %}
    ... blabla ...
    {% endcall %}

    {% call ui.render_panel("Lorem ipsum", "left") %}
    ... blabla ...
    {% endcall %}
  </div>
  <div class="right-column">
    {% call ui.render_panel("History", "left") %}
    ... blabla ...
    {% endcall %}
    {% call ui.render_panel("Now is the time for all good men", "left") %}
    ... blabla ...
    {% endcall %}
  </div>
</div>
{% endblock %}
```

No need to copy paste the same HTML code around !

Template macros

To use a macro, first import the file that contains that macro

```
{% import "ui.html" as ui %}
```

Then you can call the macro

```
{% call ui.render_panel("My Title Here", "left") %}
... blabla ...
{% endcall %}
```

What is between *call* and *endcall* could be any valid HTML code. It will be placed in place of *caller* in the macro definition.

Template language

Jinja templates use their own language, more or less Python-like.

- It tries to imitate Python
- But it is not Python

Why not having full power of Python in a template ?

Template language

Jinja provides a limited language because

- It's a view. No business code here. Just HTML generation.
- It's a page that might be served for many different users. Should be fast.

Template language

The *if* block works like Python

```
{% if show_advertisement %}
<h1>Buy Drunk Panda, the best beer in Suzhou !</h1>
{% endif %}
```

Template language

An optional *else* block works can be used

```
{% if show_advertisement %}
<h1>Buy Drunk Panda, the best beer in Suzhou !</h1>
{% else %}
Do not buy anything
{% endif %}
```

Template language

An even *elif* blocks are available

```
{% if show_beer_advertisement %}
<h1>Buy Drunk Panda, the best beer in Suzhou !</h1>
{% elif show_pizza_advertisement %}
<h1>Buy Pizza Hut, the worst pizzas ever !</h1>
{% else %}
Do not buy anything
{% endif %}
```

Template language

The Jinja *for* loop works like the Python one

```
{% for item in navigation %}
  <li>
    <a href="{{␣item.href␣}}">{{ item.caption }}</a>
  </li>
{% endfor %}
```

Note that

- *navigation* is a sequence, passed to the template
- *item* is one item of the sequence
- loop code is between {*% for %*} and {*% endfor %*}

Template language

Jinja provides a *loop* object that can be called inside a *for* loop

```
{% for item in navigation %}
  <li>
    <a href="{{ item.href }}">{{loop.index}} {{ item.caption }}</a>
  </li>
{% endfor %}
```

Template language

This *loop* object provides some useful informations about the current item of the loop

loop variable	*meaning*
loop.index	Current index (1-indexed)
loop.index0	Current index (0-indexed)
loop.revindex	Current index, reversed order (1-indexed)
loop.revindex0	Current index, reversed order (0-indexed)
loop.last	True if last item
loop.first	True if first item

Template language

You can filter the *for* loop, as in Python

```
{% for user in user_list if not user.is_hidden %}
  <li>
    {{ user.name }}
  </li>
{% endfor %}
```

Template language

If the sequence you iterate turns out to be empty, you can catch this case with an *else* block

```
{% for user in user_list if not user.is_hidden %}
  <li>
    {{ user.name }}
  </li>
{% else %}
  No users found !
{% endfor %}
```

Table of Contents

Requests

We can send data (HTML, JSON, XML, any kind of text), but we also need to *receive* data

- passwords
- checkboxes
- values
- ...

Requests

The HTTP protocol defines different kind of requests

- *GET* $\Rightarrow$ request to send data
- *POST* $\Rightarrow$ request to accept data

So far, we only handled *GET* requests : sending HTML data.

Requests

We can also handle *POST* requests, like this

```
from flask import request

@app.route('/login', methods = ['GET', 'POST'])
def login():
  # GET request
  if request.method == 'GET':
    return render_template('login.html')
  # POST REQUEST
  else:
    email    = request.form['email']
    password = request.form['password']

    # Check email & password
    # TODO

    return render_template('welcome.html')
```

Requests

The *request* object hold the information sent to the server

```
<form name="login" method="post" action="{{ url_for('login') }}">
  <label>Email</label>
  <input type="text" name="email" maxlength="254"/>

  <label>Password</label>
  <input type="password" name="password"/>

  <button type="submit">Enter</button>
</form>
```