Web Frameworks & Python

past, present, future

Aitor Guevara Escalante
aitorciki@gmail.com

The past :(
A need for standardization

Several frameworks exist (Zope, Webware, Twisted Web), but little usage compared to PHP, Perl, Java.

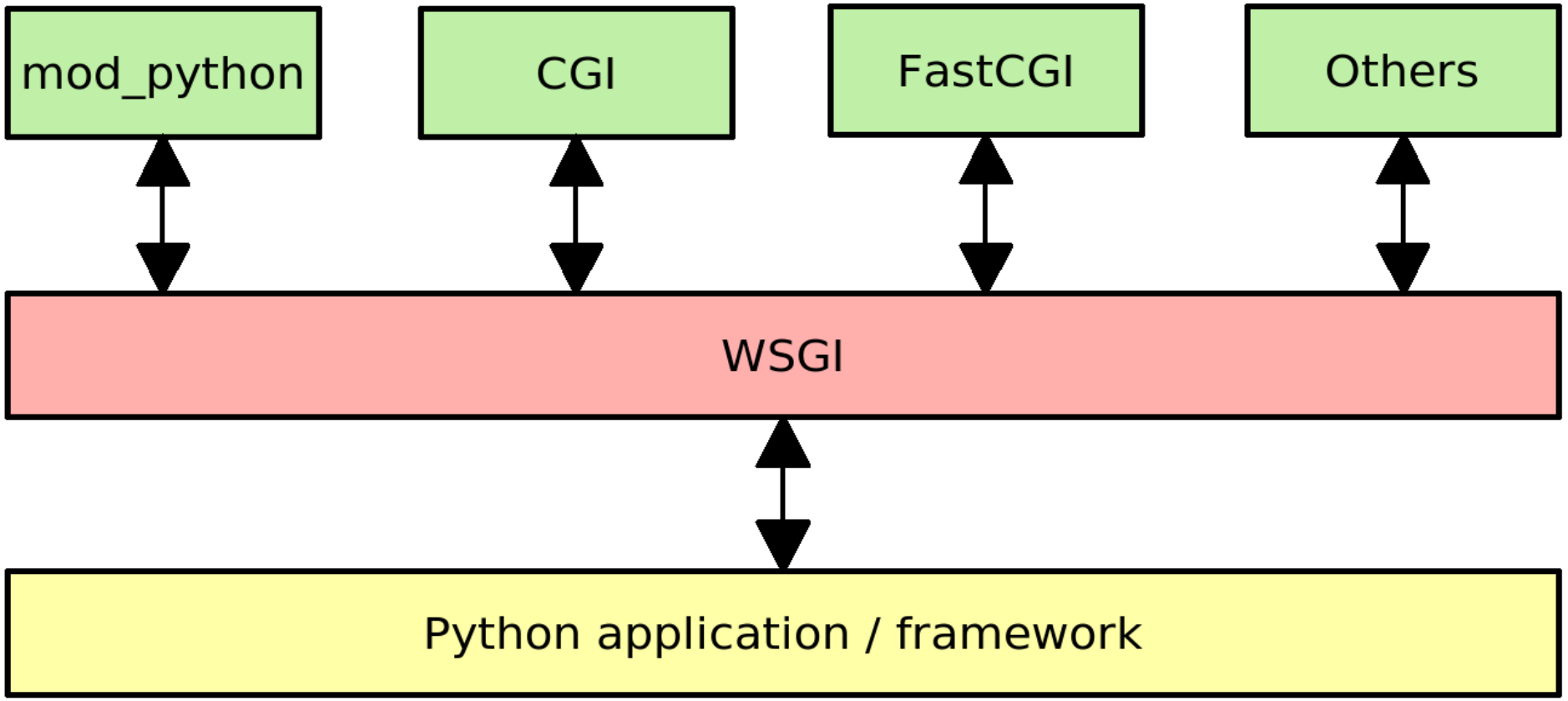
OK, but... why?

- They are bad / poor applications? No.
- Python is not known enough? No.
- Lack of standardization binds each framework to a platform? Yes.

WSGI
Web Server Gateway Interface

- PEP 333
- Standardizes interface between web servers and Python applications.
- Based on CGI standard.
- Server (gateway) object invokes callable application (framework) object.
- Middleware component acts as server and application objects:
    - URL based routing.
    - Multiple applications running in the same process.
    - Load balancing + remote processing.
    - Request preprocessing + response postprocessing.

WSGI
Web Server Gateway Interface



The present :)
Agile frameworks explosion

- Multiple new web frameworks embracing agile development appear.
- Python emerges as a top notch web language.
- Wide adoption grows, language of election in big projects / companies (Youtube, Google).
- Python enabled hostings timidly appear.
- Thanks Ruby on Rails for showing the path to the web developers community ;)

Web frameworks: Simplifying web development

Web development implies some repetitive and boring tasks. Web frameworks focus on alleviate this overhead by providing tools and libraries to easily deal with them.

- Database access and manipulation.
- Templating system.
- Session management.
- URL dispatching.
- Query parsing.

Web frameworks:
Trying to sort things

Lots of Python web frameworks exist at the moment. The following classification is based on the frameworks list found at http://wiki.python.org/moin/WebFrameworks .

- Non Full-Stack frameworks.
  Provide some (not all) of the features of a web framework as libraries.

- Full-Stack frameworks based on third party components.
  Cover the whole list of features by shipping together existing libraries.

- Full-Stack frameworks based on their own components.
  Same as the previous one, but using their own libraries written from scratch.

Non Full-Stack
Paste

http://pythonpaste.org/

Acts as WSGI wrapper to provide web development tools. Lower level than other commented frameworks and used by them.

- HTTP server.
- Request dispatching (cascading, subrequests).
- Web development (request, responses, files serving, proxying).
- Authentication framework (cookies, OpenID, HTTPAuth).
- Testing and debugging (fixtures, profiling).

Non Full-Stack CherryPy

http://www.cherrypy.org/

Higher level than Paste, more focused on easy final web application development.
Doesn't provide database access or templating system.

- HTTP server.
- Caching, encoding, static content.
- Sessions and authorization.
- Testing, profiling, coverage.
- Extensible via plugins, highly customizable.

Full-Stack + 3rd party

Pylons

http://pylonshq.com/

Aims to provide a very flexible web framework. Supports multiple modules for each functionality.

- Build on top of Paste.
- ORM: SQLAlchemy, SQLObject or DB-API.
- Uses Buffet to support lots of different template engines.
- AJAX helpers for multiple Javascript libraries.
- URL dispatching using Routes.

Full-Stack + 3rd party TurboGears

http://turbogears.org/

Focuses on less components than CherryPy (one per feature) aiming at integrating them better.

- SQLObject as ORM.
- Kid as the template system.
- CherryPy for URL handling, sessions, authorization, …
- MochiKit as Javascript library.

Full-Stack + own libraries

Zope

http://www.zope.org/

Very mature web framework. The very well known Plone CMS is built on top of it.

- Powerful multi-protocol built-in server (ZServer).
- Zope Core provides web ORB but also a search engine, a security layer.
- ZODB is Zope's own object database.
- Can be extended via ZClasses (created through web interface) and Zope Products, written in Python.

Full-Stack + own libraries

Django

http://www.djangoproject.com/

Initially written for a high-traffic news website, aims at providing a very fast, yet extremely easy to use framework. Since the existing web technologies existing at the moment didn't fulfil the creators needs, everything is written from scratch.

- Django ORM very close in concept to SQLObject.
- Very simple, as much as possible logic-free template system.
- Automatic and dynamic administration interface.
- Focused on high traffic applications, supporting out of the box multiple caching systems.
- Very active community.

The future :O
Only good stuff can happen!

- Python web frameworks adoption is growing everyday.
- Google's choice of Python for their App Engine will quickly multiply the number of running Python web applications.
- Django will have a 1.0 release very soon attracting more developers and companies.
- Good practices in web development are becoming common finally.
- Who said continuations?

Thanks for listening!

See you soon!