SVIS™

Short story

- Web service framework
- Core written in C, but controlled by Python
- Aiming for performance and scalability
- Simple
- Using existing technologies (FastCGI, Elixir/ SQLAlchemy, Mako)

Why?

- No existing software doing the same thing
- I like programming, so why not?

Smisk is ***not*** designed for HTML-based websites

It’s designed for web services in general

Practical use

You need...

- Python 2.4 or later (2.5 recommended)
- Smisk
- HTTP server (with FCGI capabilities)

Getting Smisk

- Debian packages from http://hunch.se/debian/
- PyPI/Cheesestop
- Building from source

- Also available from the Spotify Debian repository

Installing from source

First, you need to have *Python headers* and *libfcgi headers*.

```
$ curl -o smisk-1.1.0.tgz \
  http://hg.hunch.se/smisk/archive/1.1.0.tar.gz
$ tar xzf smisk-1.1.0.tgz
$ cd smisk-1.1.0
$ ./setup.py build && sudo ./setup.py install
```

LigHTTPd

```
$ cd smisk/examples/core/minimal
$ lighttpd -Df lighttpd.conf
2008-12-19 14:51:11: (log.c.75) server started

$ curl -i localhost:8080
HTTP/1.1 200 OK
Transfer-Encoding: chunked
Server: lighttpd/1.4.18 smisk/1.1.0
Date: Fri, 19 Dec 2008 13:54:21 GMT

Hello World!
```

smisk.core

Core functionality

- FastCGI server
- HTTP transaction handling
- Sessions
- Multi-processing

Simple example

```
from smisk.core import Application

class MyApp(Application):
  def service(self):
    self.response.headers = ["Content-Type: text/plain"]
    self.response("Hello World!")

MyApp().run()
```

Simple example (response)

```
$ curl -i localhost:8080/
HTTP/1.1 200 OK
Content-Type: text/plain
Transfer-Encoding: chunked
Date: Thu, 18 Dec 2008 12:19:47 GMT
Server: lighttpd/1.4.19 smisk/1.1.0

Hello World!
```

smisk.mvc

Extended functionality

- Content Negotiation
- Class tree-to-URL mapping
- Offers a Model-View-Controller design pattern
- Configuration and deployment
- Data serialisation

Content Negotiation

Client

Give me /kittens

I’m a handsome Client in my mid-30s.
I like application/plist+xml, application/json,
and prefer Latin-1 for text.

Server *(Smisk)*

Client

Here you go. /kittens.plist

I encoded the text using Latin-1, as requested.

Enjoy your kittens!

Server *(Smisk)*

Simple example

```
from smisk.mvc import *

class root(Controller):
  value = "Hello World"

  def __call__(self):
    return {"value": self.value}

  get = __call__

  def set(self, value):
    self.value = value

if __name__ == "__main__":
  main(config="my-app")
```

Simple example

```
$ curl -H 'Accept: text/plain' localhost:8080/
value: Hello World
$ curl localhost:8080/set?value=Goodbye
$ curl -H 'Accept: text/plain' localhost:8080/
value: Goodbye
```

```
$ curl -H 'Content-Type: application/json' \
  -d '{"value":[1, 2.7, 3, "midget"]}' \
  localhost:8080/set
$ curl localhost:8080/get.py
{'value': [1, 2.7000000000000002, 3, 'midget']}
$ curl localhost:8080/get.phpser
a:1:{s:5:"value";a:4:{i:0;i:1;i:1;d:2.7;i:2;i:3;i:3;s:
6:"midget";}}
```

URL-to-class tree mapping

- The *"controller tree"* – a class hierarchy.
- Canonical path mapped to leafs of the controller tree.
- Filters may remap, or rewrite, requested URLs to canonical paths (leafs).

Security

- Leafs defined outside the root of the class tree can not be accessed.
- Leafs can be explicitly hidden, or closed, using the `@hide` decorator.
- Implicit leafs can be disabled or removed (for instance the built-in external API reflection leafs).

Example

```
class root(Controller):
  def __call__(self):
    return "from __call__"

  def hej(self, value):
    return "from hej"

  @expose("hello-mom")
  def hello_mom(self, value):
    return "from hello_mom"

class kitten(root):
  def __call__(self):
    return "from kitten.__call__"

  def stroke(self, value):
    return "from kitten.stroke"
```

```
$ curl localhost/
from __call__
$ curl localhost/hej
from hej
$ curl localhost/hello-mom
from hello_mom
$ curl localhost/kitten
from kitten.__call__
$ curl localhost/kitten/__call__
404 Not Found
$ curl localhost/kitten/stroke
from kitten.stroke
```

smisk.config

Universal configuration

- JSON configuration files
- Hooks for acting on (updated) configuration
- Logging configuration built-in
- Inheritance and chaining
- Interface very similar to Spotify utils config

Example

```
"interval": 12.7, /* A value in the range [0,123) */
"kittens": ["Fatso", "Flash Gordon", "Polly"]
```

```
>>> from smisk.mvc import *
>>> from smisk.config import config
>>> config("my-app")
>>> print config["some-key"]
["Fatso", "Flash Gordon", "Polly"]
```

Learn more

- Extensive documentation: http://trac.hunch.se/smisk/docs/1.1/library/smisk.config.html
- Smisk example applications use this

smisk.wsgi

Web Server Gateway Interface

- Smisk can be used to speed up any WSGI-based service. For instance Django and Trac.
- PEP 333
- Slower than smisk.core and smisk.mvc

Simple example

```
def my_app(env, start_response):
  start_response(
    "200 OK", [("Content-Type", "text/plain")])
  return ["Hello, World!\n"]

from smisk.wsgi import main
main(my_app)
```

python-smisk.org

An open source project licensed under MIT