Zen of CherryPy

Robert E Brewer
YouGovPolimetrix

PyCon 2010

Zen of CherryPy

Koans

Intuitional, not rational

Zen of CherryPy

Ikkyu asked Subhuti,

Zen of CherryPy

Ikkyu asked Subhuti,

how do I turn on gzip?

Zen of CherryPy

Ikkyu asked Subhuti,

how do I turn on gzip?

Subhuti answered,

Zen of CherryPy

Ikkyu asked Subhuti,

how do I turn on gzip?

Subhuti answered,

tools.gzip.on

Zen of CherryPy

```
public void doGet(
    HttpServletRequest httpServletRequest,
    HttpServletResponse httpServletResponse)
        throws IOException, ServletException{

  OutputStream out = null
  String encoding = httpServletRequest.getHeader("Accept-Encoding");

  if (encoding != null && encoding.indexOf("gzip") != -1){
      httpServletResponse.setHeader("Content-Encoding" , "gzip");
      out = new GZIPOutputStream(httpServletResponse.getOutputStream());
  }
  else if (encoding != null && encoding.indexOf("compress") != -1){
      httpServletResponse.setHeader("Content-Encoding" , "compress");
      out = new ZIPOutputStream(httpServletResponse.getOutputStream());
  }
  else{
      out = httpServletResponse.getOutputStream();

  }
}
```

Zen of CherryPy

```
<Location "/">
   SetOutputFilter DEFLATE
   AddOutputFilterByType DEFLATE
      text/html text/plain
   # Don't compress images
   SetEnvIfNoCase Request_URI
      \.(?:gif|jpe?g|png)$ no-gzip
      dont-vary
</Location>
```

Zen of CherryPy

```
[/]
tools.gzip.on: True
```

Zen of CherryPy

One

Common tasks should be
fast and easy

Zen of CherryPy

Ikkyu asked Subhuti,

Zen of CherryPy

Ikkyu asked Subhuti,

how do I turn off sessions?

Zen of CherryPy

Ikkyu asked Subhuti,

how do I turn off sessions?

Subhuti answered,

Zen of CherryPy

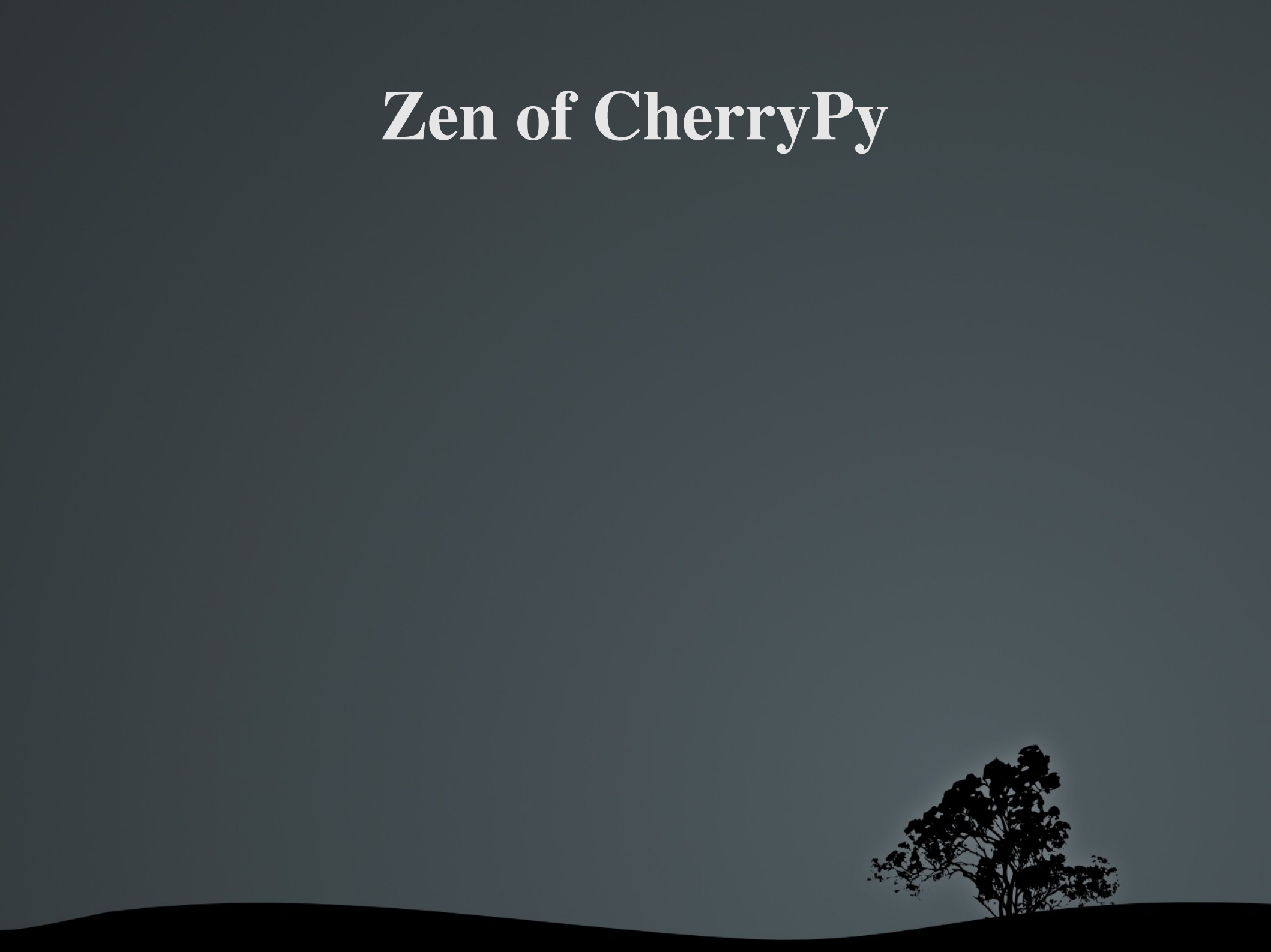
1. Click Start,
   point to Programs,
   click Administrative Tools, and then
   click Internet Information Services.
2. Right-click your Web site, and then
   click Properties.
3. Click the Home Directory tab.
4. Click Configuration, and then
   click the Options tab.
5. Click to clear the Enable Session State check box.

Zen of CherryPy

Locate the relevant Context element in either server.xml or in the context descriptor and declare the no-op session manager.

```
<Context .. >
  <Manager pathname="" />
</Context>
```

Zen of CherryPy



Zen of CherryPy

One

Common tasks should be
fast and easy

Zen of CherryPy

One

Common tasks should be
fast and easy

Two

Doing nothing should be
easier and faster

Zen of CherryPy

Ikkyu asked Subhuti,

Zen of CherryPy

Ikkyu asked Subhuti,

how do I turn on Subhuti?

Zen of CherryPy

Ikkyu asked Subhuti,

how do I turn on Subhuti?

Subhuti answered,

Zen of CherryPy

Ikkyu asked Subhuti,

how do I turn on Subhuti?

Subhuti answered,

tools.subhuti.on

Zen of CherryPy

Ikkyu asked Subhuti,

how do I turn on Subhuti?

Subhuti answered,

tools.subhuti.on

From that day on, Ikkyu asked Subhuti no more questions

Zen of CherryPy

jquery/src/effects.js

```
jQuery.fn.extend({
  fadeTo:   function( speed, to, callback ) {
    return this.
      .filter(":hidden")
      .css("opacity", 0)
      .show().end()
      .animate({opacity: to}, speed, callback);

  }
});
```

Zen of CherryPy

http://scrunchup.com/issue-2/the-evolution-of-a-jquery-plugin/

```
jQuery.fn.extend({
  slideOut: function( ) {
    return this.each(function(speed) {
      this.data('oldwidth', this.css('width'))
          .css({width: '0px'})
          .animate({width:this.data('oldwidth')},
                   (speed ? speed : 500));
    });
  }
});
```

Zen of CherryPy

cherrypy/lib/cptools.py

```
def log_headers():
    h = ["  %s: %s" % (k, v)
         for k, v in request.header_list]
    log('Headers:\n' + '\n'.join(h))
tools.log_headers =
    Tool('before_error_response', log_headers)

class Root:
    @tools.log_headers()
    def services(self, *args, **kw):
        ...
```

Zen of CherryPy

http://code.google.com/p/pyjamas/wiki/
PyjamasWithCherryPyJSONRPC

```
def jsonrpchdl():
    json_string = request.body.read()
    obj = json.loads(json_string)
    request.myparams = obj
tools.jsonrpchdl =
    Tool('before_handler', jsonrpchdl)

class Root:
    @tools.jsonrpchdl()
    def services(self, *args, **kw):
        ...
```

Zen of CherryPy

```
class Response(object):

  def encode_content(self, encoding='gzip'):
    """
    Encode the content with the given
    encoding (only gzip and identity
    are supported).
    """
```

Zen of CherryPy

Three

Extensions to CherryPy
should look like CherryPy

Zen of CherryPy

Three

Extensions to CherryPy
should look like CherryPy
should look like
Extensions to CherryPy

Zen of CherryPy

Joshu asked,

Zen of CherryPy

Joshu asked,

may I please have a server?

Zen of CherryPy

Joshu asked,

may I please have a server?

His master replied,

Zen of CherryPy

Joshu asked,

may I please have a server?

His master replied,

"server"

Zen of CherryPy

Dispatcher

/thing?foo=1&bar=2 -> Root.thing(foo, bar)

```
class Root:

  def thing(self, foo, bar):
    method = cherrypy.request.method
    if method == 'GET':
      ...
    elif method == 'POST':
      ...
```

Zen of CherryPy

MethodDispatcher

/thing?foo=1&bar=2 -> Thing.GET(foo, bar)

```
class Thing:

  def GET(self, foo):
     ...

  def POST(self, bar):
     ...
```

Zen of CherryPy

```
>>> dispatchers["method"] =
        cherrypy.dispatch.MethodDispatcher
```

```
[/]
request.dispatch: "method"
```

```
>>> dispatchers[request.dispatch]().run()
```

Zen of CherryPy

```
[/]
request.dispatch:
    cherrypy.dispatch.MethodDispatcher(
        foo, bar)
```

```
>>> request.dispatch.run()
```

Zen of CherryPy

Four

Objects are better than referents

Zen of CherryPy

The master asked Kakua,

Zen of CherryPy

The master asked Kakua,

what is CherryPy?

Zen of CherryPy

The master asked Kakua,

what is CherryPy?

Kakua answered,

Zen of CherryPy

The master asked Kakua,

what is CherryPy?

Kakua answered,

CherryPy is a wrapper for
HTTP, XML, HTML, REST,
AJAX, CSS, JS, RSS and JSON

Zen of CherryPy

The master asked Kakua,

what is CherryPy?

Kakua answered,

CherryPy is a wrapper for
HTTP, XML, HTML, REST,
AJAX, CSS, JS, RSS and JSON

The master replied,

Zen of CherryPy

The master asked Kakua,

what is CherryPy?

Kakua answered,

CherryPy is a wrapper for
HTTP, XML, HTML, REST,
AJAX, CSS, JS, RSS and JSON

The master replied,

you are gravely
mistaken...

Zen of CherryPy

CherryPy is an object,
with attributes and methods

Zen of CherryPy

```
get '/:name' do
  content_type 'text/xml', :charset => 'utf-8'
  "<root><h1>Hello #{params[:name]}</h1></root>"
end
```

Zen of CherryPy

```
get '/:name' do
  content_type 'text/xml', :charset => 'utf-8'
  "<root><h1>Hello #{params[:name]}</h1></root>"
end
```

How many of us know HTTP?

Zen of CherryPy

```
get '/:name' do
  content_type 'text/xml', :charset => 'utf-8'
  "<root><h1>Hello #{params[:name]}</h1></root>"
end
```

How many of us know HTTP?
How many of us understand the above?

Zen of CherryPy

How many of us know HTTP?
How many of us understand the above?
Or could write it without an example?

Zen of CherryPy

```
get '/:name' do
  content_type 'text/xml', :charset => 'utf-8'
  "<root><h1>Hello #{params[:name]}</h1></root>"
end
```

How many of us know HTTP?
How many of us understand the above?
Or could figure out how to emit the "X-Powered-By" response header?

Zen of CherryPy

```
get '/:name' do
  content_type 'text/xml', :charset => 'utf-8'
  "<root><h1>Hello #{params[:name]}</h1></root>"
end
```

How many of us know HTTP?
How many of us understand the above?
Or could handle the "OPTIONS" HTTP method?

Zen of CherryPy

```
import cherrypy

class Thing:

  @cherrypy.expose
  def GET(self, name=None):
    cherrypy.response.headers['Content-Type'] =
      'text/xml;charset=utf-8'
    return "<root><h1>Hello %s</h1></root>" %
      name
```

Zen of CherryPy

Five

Domain-specific Python
is better than new DSL's

Zen of CherryPy

The master asked Kakua again,

Zen of CherryPy

The master asked Kakua again,

what is CherryPy?

Zen of CherryPy

The master asked Kakua again,

what is CherryPy?

Kakua answered,

Zen of CherryPy

The master asked Kakua again,

what is CherryPy?

Kakua answered,

CherryPy is an object,
with attributes and methods

Zen of CherryPy

The master asked Kakua again,

what is CherryPy?

Kakua answered,

CherryPy is an object,
with attributes and methods

Angrily, the master answered,

no!

Zen of CherryPy

CherryPy is a function,
with inputs and outputs

Zen of CherryPy

```
<database>
  <jndi-name>jdbc/name</jndi-name>
  <driver>
    <type>org.postgresql.Driver</type>
    <url>jdbc:postgresql://
         127.0.0.1:5432/dbname</url>
    <user>username</user>
    <password>password</password>
  </driver>
  ...
</database>
```

Zen of CherryPy

```
from sqlalchemy.schema import MetaData

metadata = MetaData(
    'postgres://username:password'
    '@127.0.0.1:5432/dbname',
    echo=False,
    max_overflow=15)
```

Zen of CherryPy

```
import couchdb
couch = couchdb.Server()
db = couch.create(dbname)
```

Zen of CherryPy

Five

Domain-specific Python
is better than new DSL's

Zen of CherryPy

Five

Domain-specific Python
is better than new DSL's

Six

But imperative Python
is better yet

Zen of CherryPy

The master asked Kakua a third time,

Zen of CherryPy

The master asked Kakua a third time,

what is CherryPy?

Zen of CherryPy

The master asked Kakua a third time,

what is CherryPy?

Kakua answered,

Zen of CherryPy

The master asked Kakua a third time,

what is CherryPy?

Kakua answered,

CherryPy is a function
and an object

Zen of CherryPy

The master asked Kakua a third time,

what is CherryPy?

Kakua answered,

CherryPy is a function
and an object

The master answered,

Zen of CherryPy

CherryPy
is not made for CherryPy,
but it is made from CherryPy

Zen of CherryPy

CherryPy
is not made for CherryPy,
but it is made from CherryPy

At that moment, Kakua was enlightened

Zen of CherryPy

```
from cherrypy.lib.cptools import accept

class CSS:

  @cherrypy.expose
  def here_css(self):
    accept('text/css')
    return "div {font-size: 110%}"
```

Zen of CherryPy

```
class CSS:
  @tools.accept('text/css')
  @cherrypy.expose
  def here_css(self):
    return "div {font-size: 110%}"
```

Zen of CherryPy

```
class CSS:

  @cherrypy.expose
  def here_css(self):
    return "div {font-size: 110%}"
```

```
[/css]
tools.accept.on: True
tools.accept.media: 'text/css'
```

Zen of CherryPy

Five

Domain-specific Python is better than new DSL's

Zen of CherryPy

Five

Domain-specific Python is better than new DSL's

Six

But imperative Python is better yet

Zen of CherryPy

Five

Domain-specific Python is better than new DSL's

Six

But imperative Python is better yet

Seven

Unless that isn't enough

Zen of CherryPy

Gasan told his followers,

Zen of CherryPy

Gasan told his followers,

request headers should be
a mapping of unique names to multiple values,
with methods for inspecting and manipulating
each collection singly or in groups

Zen of CherryPy

Gasan told his followers,

request headers should be
a mapping of unique names to multiple values,
with methods for inspecting and manipulating
each collection singly or in groups

One of his acolytes murmured,

Zen of CherryPy

Gasan told his followers,

request headers should be
a mapping of unique names to multiple values,
with methods for inspecting and manipulating
each collection singly or in groups

One of his acolytes murmured,

a dict does most of that

Zen of CherryPy

Gasan told his followers,

request headers should be
a mapping of unique names to multiple values,
with methods for inspecting and manipulating
each collection singly or in groups

One of his acolytes murmured,

a dict does most of that

Gasan shouted, *Excellent!*
You are not far from CherryPy-nature

Zen of CherryPy

```
class HeaderMap(object):

  def __init__(self):
    self._map = {}

  def set(self, key, value):
    self._map[str(key).title()] = value

  def get_all(self):
    return self._map.items()
```

Zen of CherryPy

```
class CaseInsensitiveDict(dict):

  def __setitem__(self, key, value):
    dict.__setitem__(
      self, str(key).title(), value)


class HeaderMap(CaseInsensitiveDict):
  ...
```

Zen of CherryPy

On Understanding Data Abstraction, Revisited
http://www.cs.utexas.edu/~wcook/Drafts/2009/essay.pdf

- Abstract Data Types (int, set) = type abstraction
  - Facilitate adding new operations:
    `merge(headers, headers) -> headers`
    `elements(headers) -> *HeaderElement`
- Objects = procedural abstraction
  - Facilitate adding new representations:
    `AcceptHeaders, TokenHeaders, TEXTHeaders, DateHeaders, HTTP11Headers`

Zen of CherryPy

Eight

Subclassed builtins
are better than
custom types

Zen of CherryPy

Nine

But builtin types
are even better

Zen of CherryPy

{ }

Zen of CherryPy

A student boasted to Hakuin,
each application I write
has a better templating engine
than the previous one

Zen of CherryPy

A student boasted to Hakuin,
each application I write
has a better templating engine
than the previous one

Hakuin said,
Good! I need you to write an application for me

Zen of CherryPy

A student boasted to Hakuin,
each application I write
has a better templating engine
than the previous one

Hakuin said,
Good! I need you to write an application for me

The student (who was very experienced) asked,

Zen of CherryPy

A student boasted to Hakuin,
each application I write
has a better templating engine
than the previous one

Hakuin said,
Good! I need you to write an application for me

The student (who was very experienced) asked,
aha, and what will you be producing,
HTML or XML?

Zen of CherryPy

A student boasted to Hakuin,
each application I write
has a better templating engine
than the previous one

Hakuin said,
Good! I need you to write an application for me

The student (who was very experienced) asked,
aha, and what will you be producing,
HTML or XML?

Hakuin replied,

Zen of CherryPy

A student boasted to Hakuin,
each application I write
has a better templating engine
than the previous one

Hakuin said,
Good! I need you to write an application for me

The student (who was very experienced) asked,
aha, and what will you be producing,
HTML or XML?

Hakuin replied,
yes

Zen of CherryPy

HTML or XML?

yes

Zen of CherryPy

Zero-One-Infinity Rule

http://catb.org/jargon/html/Z/Zero-One-Infinity-Rule.html

- 0 - disallow the entity entirely,
- 1 - allow exactly one “exception”, or
- ∞ - allow as many as the user wants

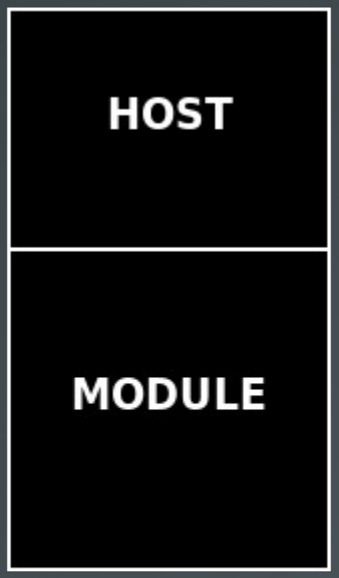
Zen of CherryPy

$1 < \infty$

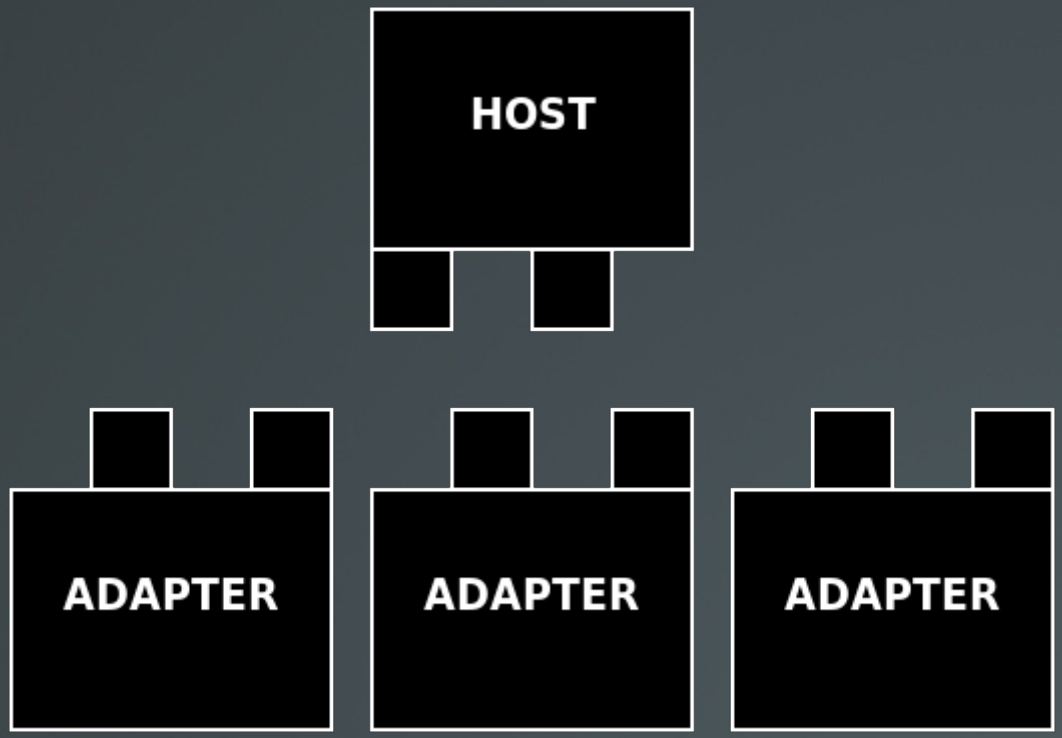
but

$0 > 1$

Zen of CherryPy



Zen of CherryPy



Zen of CherryPy

Zen of CherryPy

$1 < \infty$

but

$0 > 1$

Zen of CherryPy

Ten

1 < 2

but

0 > 1

Zen of CherryPy

www.cherrypy.org