Python

Desenvolvendo uma Aplicação WEB usando o CherryPY

Antonio Sérgio Nogueira

Presidente Prudente
2009

Python

CherryPy

1.Introdução: Como definido em seu manual, o CherryPy é uma simples e poderosa biblioteca Python para os desenvolvedores web que desejam encontrar um pacote que esconde as dificuldades do protocolo HTTP mantendo seus pontos fortes. HTTP é a espinha dorsal da World Wide Web. Aplicações Web tem crescido exponencialmente nos últimos anos. Esta explosão foi seguida por um grande número de ferramentas, bibliotecas e frameworks em várias linguagens de programação para ajudar os desenvolvedores web em sua tarefa. Em última análise, todos estas ferramentas visam tornar a vida de um desenvolvedor web muito mais fácil. Neste contexto CherryPy começou a usar os pontos fortes do Python como uma linguagem dinâmica para modelar e vincular o protocolo HTTP em uma Interface de Programa de Aplicação que segue expressões Python.

2.Requisitos: para usar esta biblioteca você precisa instalar:

- Python 2.4 ou versão mais recente (linguagem)
- CherryPy 3.1.2 (biblioteca)

3. Pontos fortes do CherryPy: simplicidade, pacote auto-contido(só necessita do Python),não intrusivo (o desenvolvedor não precisa ficar fazendo suposições sobre o modo de uso da biblioteca), aberto a discussão(os desenvolvedores aceitam sugestões).

4. Instalando o CherryPy no Windows: para instalar o CherryPy basta fazer um download da biblioteca CherryPy-3.1.2.win32.exe em cherry.org. Depois basta executar o software e definir o diretório onde está o Python para instalar a biblioteca. Outras versões e instalações vide referência bibliográfica. Após instalar o software execute o ide do Python e digite o comando:

```
>>>import cherrypy
>>>cherrypy.__version__
'3.1.2'
```

Pronto o software esta instalado, se acontecer algum erro aparece a mensagem:

```
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    import cherrypy
ImportError: No module named cherrypy
```

5. Algumas Definições:

Palavra Chave	Definição
Servidor Web	Um servidor web é uma interface que lida com o protocolo HTTP. Seu objetivo é transformar pedidos HTTP em entidades que são passadas para o servidor de aplicação e então transformar a informação retornada pelo servidor de aplicação em uma resposta HTTP.
Aplicação	É um pedaço de software que pega uma unidade de informação e aplica nela a regra do negócio e retorna uma unidade processada de informação.
Servidor de Aplicação	É um componente que hospeda uma ou mais aplicações.
Servidor de Aplicação WEB	Um servidor de aplicação web é simplesmente a junção de um servidor web com um servidor de aplicação em um componente.

CherryPy Um servidor de aplicação web.

6. Desenvolvendo Aplicações: o desenvolvimento Web é uma tarefa na qual o Python não é popular, mas existem interessantes soluções para desenvolvimento Web. O CherryPy facilita o desenvolvimento de aplicações Web. O CherryPy roda no seu próprio servidor, mas é fácil executá-lo no Apache. Os requisitos são mapeados em objetos dentro da aplicação e são executados.

Como sempre o programa: AloMundo.py

```
# -*- coding: iso-8859-1 -*-   # Alfabeto Latino
import cherrypy                 # carregando biblioteca

class HelloWorld:
   def index(self):
      return "Alô Mundo"
   index.exposed = True         #  permite definir quais métodos serão expostos na Web

cherrypy.quickstart(HelloWorld())  #publica instância da classe e inicia servidor
```

Quando a aplicação é executada, o servidor é iniciado com configuração default(ele ouvirá a porta 8080 na hospedagem local, essa configuração pode ser substituída). Quando uma requisição é recebida pela URI http://localhost:8080, neste caso particular o site raiz é mapeado pelo método index(). A classe HelloWord define um método index que será exposto e o servidor CherryPy chama HelloWorld().index() e o resultado é enviado para o browser como o conteúdo da página index para o website(similar ao index.html que é a página padrão para servidores Web convencionais).

6.1 Conceitos:

Objetos de Publicação: qualquer objeto agregado ao objeto root é dito objeto publicado, isto significa que o objeto é acessível via rotina interna de mapeamento do objeto URL. Entretanto, não significa que o objeto é diretamente acessível via Web, para que isto ocorra o objeto deve ser exposto.

Expondo Objetos: o cherrypy mapeia as requisições da URL para objetos e invoca automaticamente a chamada adequada. Esta chamada adequada pode ser invocada como resultado de uma requisição externa que pede para o objeto ser exposto. A exposição de objetos em cherrypy são permitidos pelo atributo ***exposed,*** que pode ser diretamente setados para objetos que são chamados.

Expondo objetos(setando diretamente o atributo exposed):

```
class Root:
    def index(self):
        ...
    index.exposed = True
```

ou usando decorator:

```
@cherrypy.expose

    def index(self):
```

```
        ...
```

Para métodos especiais como __call__ , veja o exemplo a seguir:

```
class Node:

      exposed=True

      def __call__(self):

          ...
```

Teoricamente todos os objetos expostos são publicados e devemos então ter cuidado com objetos não expostos.

6.2 Encontrando o objeto correto: ao digitar uma URI o usuário recebe a página desejada. Um servidor Web convencional usa a URI para recuperar um arquivo estático de um sistema de arquivos. Por outro lado servidores Web podem além de fornecer arquivos estáticos, mapear objetos para a URI de forma a receber e chamar alguns objetos. Como resultado o browser recebe uma página que é o resultado de uma aplicação dinâmica, para cada URI apenas um objeto pode ser chamado. Entendendo o mapeamento feito pelo CherryPy que é bastante simples, permite-nos desenvolver aplicações. Ao receber uma URI, o CherryPy reparte ela em componentes de caminho(path) e procura um objeto no site que é a melhor resposta para esta URI particular. Para cada componente de caminho ele tenta encontrar um objeto de mesmo nome, começando pela raiz(root) e vai descendo por cada componente até encontrá-lo, vejamos:

```
root = HelloWorld()
root.onepage = OnePage()
root.otherpage = OtherPage()
```

Neste exemplo a URI http://localhost:8080/onepage irá apontar para o primeiro objeto e http://localhost:8080/otherpage o segundo objeto e mais longe ainda vejamos:

```
root.some = Page()
root.some.page = Page()
```

Neste exemplo http://localhost:8080/some/page irá mapear o objeto root.some.page que será chamado se o mesmo estiver exposto.
No programa AloMundo.py para adicionamos onepage, twopage, threepage fazemos o seguinte:

```
#-*-coding:iso-8859-1-*-
import cherrypy
class TwoPage(object):
   def index(self):
      return "página2"
   index.exposed=True

class OnePage(object):
   threepage=TwoPage()
   def index(self):
      return "Página Um!"
```

```
    index.exposed=True

class HelloWorld(object):
    onepage = OnePage()
    twopage=TwoPage()


    def index(self):
        return "Alô Mundo"
    index.exposed=True

    def foo(self):
        return "foo"
    foo.exposed=True
cherrypy.quickstart(HelloWorld())
```

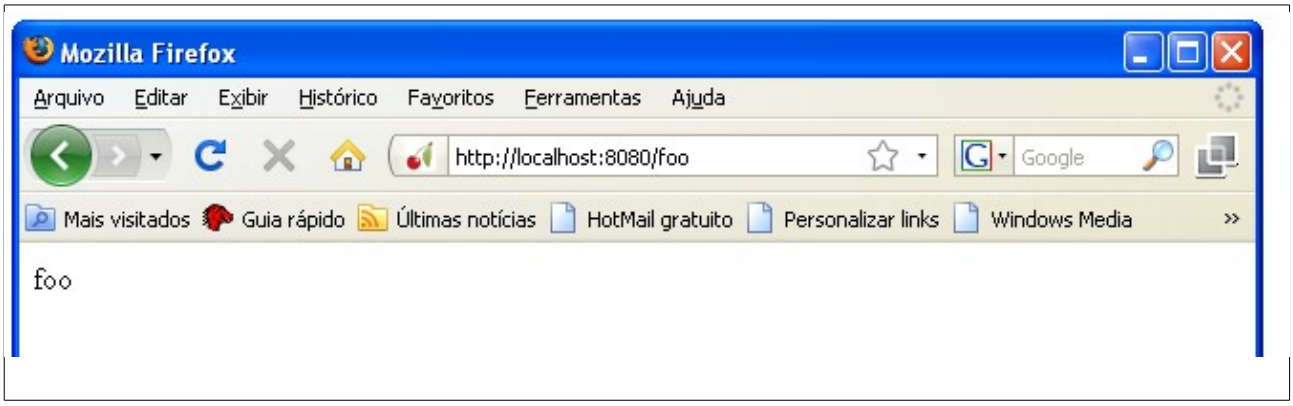
http://localhost:8080/onepage/ , http://localhost:8080/twopage/ , http://localhost:8080/onepage/threepage/

6.3 O método index: o método index() tem uma regra especial em CherryPy, parecido com o arquivo index.html, ele é a página default para qualquer nó interno na árvore de objetos. Este método pode ter argumentos adicionais que são mapeados em variáveis de formulário como as enviadas pelo métodos GET ou POST.

6.4 Chamando outros métodos: cherrypy pode chamar diretamente métodos em objetos publicados, ele recebe a URL e mapeia direto para eles.

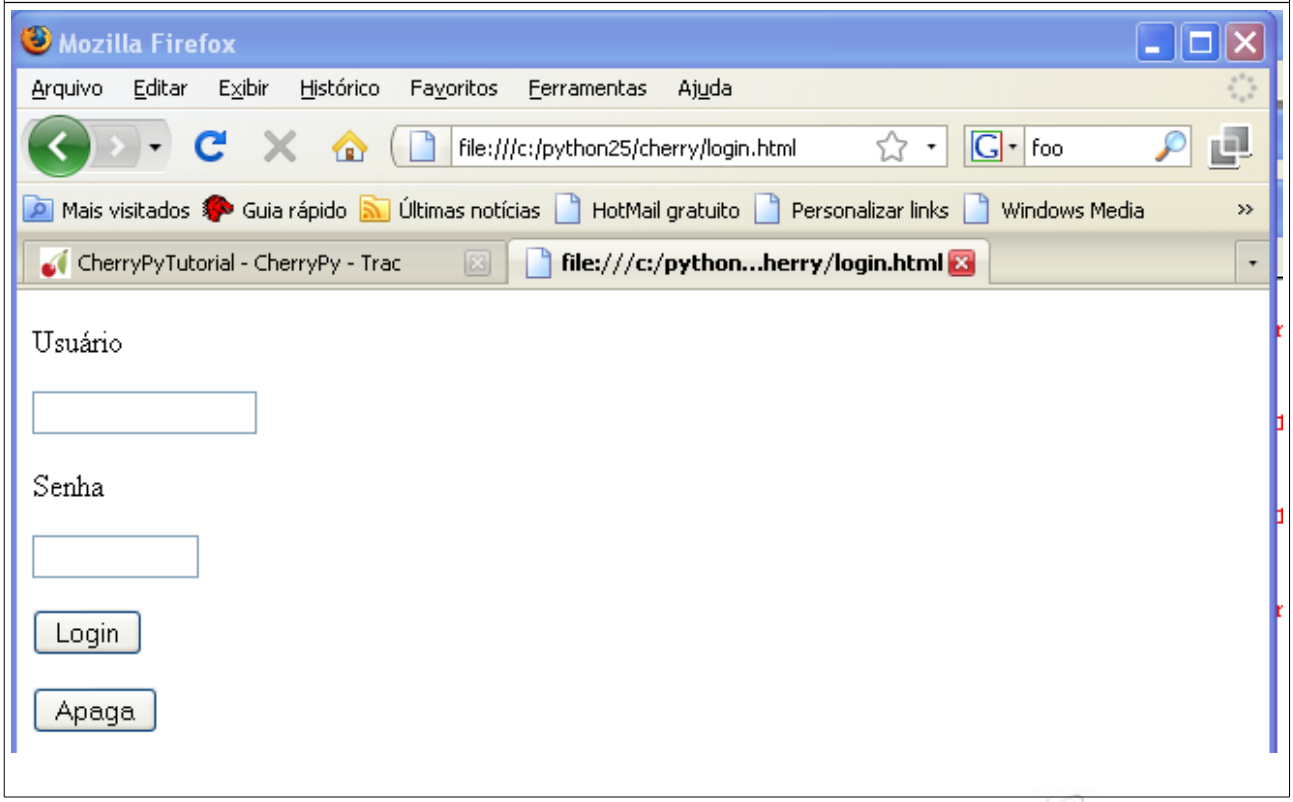
```
# -*- coding: iso-8859-1 -*-
import cherrypy
class OnePage(object):
    def index(self):
        return "Página Um!"

    index.exposed=True

class HelloWorld(object):
    onepage = OnePage()

    def index(self):
        return "Alô Mundo"
    index.exposed=True

cherrypy.quickstart(HelloWorld())
```

Execute a URL: http://localhost:8080/foo



6.5 Recebendo dados de Formulários HTML: Qualquer método pode receber dados adicionais do formulário HTML usando argumentos (keyword arguments). Vejamos um exemplo que envia dados para login usando o método POST.
Arquivo: login.html

```
<form action="doLogin" method="post">
   <p>Usuário</p>
   <input type="text" name="username" value=""
      size="15" maxlength="40"/>
   <p>Senha</p>
   <input type="password" name="password" value=""
      size="10" maxlength="40"/>
   <p><input type="submit" value="Login"/></p>
   <p><input type="reset" value="Apaga"/></p>
</form>
```



Execute o arquivo HTML com a URL file:///c:/python25/cherry/login.html e veja o resultado acima.

Arquivo: login.py

```
# -*- coding: iso-8859-1 -*-
import cherrypy
class OnePage(object):
    def index(self):
        return "Página Um!"

    index.exposed=True


class HelloWorld(object):
    onepage = OnePage()

    def index(self):
        return """<form action="doLogin" method="post">
            <p>Username</p>
            <input type="text" name="username" value=""
            size="15" maxlength="40"/>
            <p>Password</p>
            <input type="password" name="password" value=""
            size="10" maxlength="40"/>
            <p><input type="submit" value="Login"/></p>
            <p><input type="reset" value="Clear"/></p>
            </form>"""

    index.exposed=True

    def foo(self):
        return "foo"
    foo.exposed=True


    def doLogin(self,username=None,password=None):
        return 'Olá '+username+' '+password
    doLogin.exposed=True


cherrypy.quickstart(HelloWorld())
```

Neste exemplo anterior mostramos a passagem de dados, execute e veja o que acontece. Modifique a linha foo.exposed para foo.exposed=False e veja o que acontece.

6.6 Método Default e caminho parcial – pode ser utilizado para manipular erros de mapeamento da URL ou suportar argumentos posicionais.

```
#-*-coding:iso-8859-1-*-
import cherrypy

class HelloWorld(object):

    def index(self):
```

```
        return "Alô Mundo"
    index.exposed=True

    def foo(self):
        return "foo"
    foo.exposed=True

    def default(self,qqcoisa):
        return "Não tem esse caminho."
    default.exposed=True

cherrypy.quickstart(HelloWorld())
```

Experimente digitar: http://localhost:8080/x Veja que a mensagem que é emitida: ”Não tem esse caminho.”

```
# emite a data digitada como parâmetro da URL
# exemplo:URL (http://localhost/onepage/2009/12/01

class OnePage(object):

    def index(self):
        return "Página Um!"
    index.exposed=True
    def default(self,year,month,day):
        return "Data:"+year+"/"+month+"/"+day
    default.exposed=True
```

Exemplo de um programa(default.py) com passagem de parâmteros na URL:

```
# -*- coding: iso-8859-1 -*-
import cherrypy

class OnePage(object):

    def index(self):
        return "Página Um!"
    index.exposed=True
    def default(self,year,month,day):
        return "Data:"+year+"/"+month+"/"+day
    default.exposed=True


class HelloWorld(object):

#    onepage = OnePage()

    def index(self):
```

```
    return """<form action="doLogin" method="post">
        <p>Username</p>
        <input type="text" name="username" value=""
        size="15" maxlength="40"/>
        <p>Password</p>
        <input type="password" name="password" value=""
        size="10" maxlength="40"/>
        <p><input type="submit" value="Login"/></p>
        <p><input type="reset" value="Clear"/></p>
        </form>"""

  index.exposed=True

  def foo(self):
      return "foo"
  foo.exposed=True


  def doLogin(self,username=None,password=None):
      return 'Olá '+username+' '+password
  doLogin.exposed=True
```

6.7 Arquivo de configuração do CherryPy: o CherryPy usa um simples arquivo de configuração para customizar seu comportamento. Há atualmente dois tipos de arquivos, um para o site e um para cada aplicação, mas se você tiver apenas uma aplicação você pode colocar ambos num único arquivo. Por exemplo:

```
[global]
server.socket_port = 8000
server.thread_pool = 10  # quantas thread o servidor pode iniciar
tools.sessions.on = True # habilita sessão, usada sites complexos
tools.staticdir.root = "/home/site" # diretório onde estão aplic. estáticas

[/static]
tools.staticdir.on = True
tools.staticdir.dir = "static"
```

Para executar este arquivo de configuração para o site use **cherrypy.config.update(arquivo_ou_dicionário)** ou cherry.tree.mount(Root(),'/', arquivo_ou_dicionário) para a aplicação.

```
# arquivo global.cfg
[global]
server.socket_port=8000 #definindo porta
server.thread_pool=10
```

```
# arquivo defaultconfig.py
# -*- coding: iso-8859-1 -*-
import cherrypy
```

Python

```
class OnePage(object):

   def index(self):
      return "Página Um!"
   index.exposed=True
   def default(self,year,month,day):
      return "Data:"+year+"/"+month+"/"+day
   default.exposed=True


class HelloWorld(object):

#    onepage = OnePage()

   def index(self):
      return """<form action="doLogin" method="post">
         <p>Username</p>
         <input type="text" name="username" value=""
         size="15" maxlength="40"/>
         <p>Password</p>
         <input type="password" name="password" value=""
         size="10" maxlength="40"/>
         <p><input type="submit" value="Login"/></p>
         <p><input type="reset" value="Clear"/></p>
         </form>"""

   index.exposed=True

   def foo(self):
      return "foo"
   foo.exposed=True


   def doLogin(self,username=None,password=None):
      return 'Olá '+username+' '+password
   doLogin.exposed=True


hello=HelloWorld()
hello.onepage=OnePage()
cherrypy.config.update('global.cfg') # lendo arquivo global
cherrypy.quickstart(hello)
```

6.8 Estrutura do CherryPy: esta biblioteca possui vários membros.

- cherrypy.engine - interface de controle do cherrypy.
- cherrypy.server – servidor HTTP.
- cherrypy.request – contém a informação de requisição do HTTP.
- cherrypy.session – mapeamento especial gerado pelo cherrypy.
- cherrypy.response – dados para construir a resposta HTTP.

6.9 Ferramentas do Cherrypy: várias ferramentas são fornecidas como parte da biblioteca padrão do Cherrypy por ele ter um núcleo extremamente leve e limpo. Algumas delas são: tools.decode(processa automaticamente dados Unicode transformando em strings do Python), tools.encode(transforma string do Python em códigos adequados como UTF-8 para o browser) , tools.gzip(compactar dados), tools.xmlrpc(implementa XML-RPC). As ferramentas podem ser aplicadas em diferentes partes do site sem a necessidade de mudar internamente o programa, basta usar um arquivo de configuração. Veja exemplo:

```
[/]
tools.encode.on = True
tools.gzip.on = True
```

A aplicação pode usar string Unicode que a translação para UTF-8 será automática, e todo conteúdo será comprimido, economizando banda.

7.Coletânea de Aplicações:

7.1: Rodando um programa Python via net

```
#programapelanet.py
# -*- coding:UTF-8 -*-
import cherrypy
import os
from tempfile import gettempdir
TMP_DIR = gettempdir()

class PythonRunner(object):
 title = 'Rodando Programa com Python via Net'

 def header(self):
   return '''
          <html>
          <head>
             <title>%s</title>
          </head>
          <body>
          <h2>%s</h2>
       ''' % (self.title, self.title)

 def footer(self):
   return '''
          </body>
          </html>
       '''

 def index(self, code=None):

   output = ''
   if code is None:
      output='''
```

```
    else:
        tmp_filename = os.path.join(TMP_DIR, 'myfile.dat')
        f = open(tmp_filename, 'w')
        f.write(code)
        f.close()
        f_in, f_out = os.popen4("python %s"%tmp_filename)
        output = "O codigo eh:"
        output += "<pre><font color='blue'>%s</font></pre>resultando: "%code
        output += "<pre><font color='green'>"
        for line in f_out.readlines():
            output += line
        output += "</font></pre>"

    return self.header()+'''
            Digite seu codigo.
            <form action="index" method="GET">
            <textarea name="code" rows=5 cols=80></textarea><br/>
            <input type="submit" value="Execute em Python"/>
            </form>
            <br/>
            %s
        ''' % output + self.footer()
 index.exposed = True

cherrypy.root = PythonRunner()


if __name__ == '__main__':
    cherrypy.quickstart(cherrypy.root)
```

7.2 Aplicação Nota – permite que o usuário deixe um recado na página.

```
# -*-coding:iso-8859-1-*-
import cherrypy
import time
notas=[]
cabecalho="""
    <html>
        <head>
            <title>Notas</title>
        </head>
        <body>"""
rodape="""
        </body>
    </html>"""

form_nota="""
        <class="form">
        <form method="post" action="post" class="form">
```

```
        <input type="text" value="Sua nota aqui...." name="text"
           size="60"></input>
        <input type="submit" value="Adicionar"></input>
      </form>"""

vista_nota="""
        %s
        <class="info">%s<p>"""

form_autor="""
      <class="form">
      <form method="post" action="set">
        <input type="text" name="name"></input>
        <input type="submit" value="Registra autor"</input>
      </form>"""

class Autor(object):
   @cherrypy.expose
   def index(self):
      return[cabecalho,form_autor,rodape]
   @cherrypy.expose
   def set(self,name):
      cherrypy.session['autor']=name
      cherrypy.HTTPRedirect('/')
         return[cabecalho,"""Oi %s. Deixe a mensagem <a href="/" title="Home">aqui.</a>"""%
(name,),rodape]

class Nota(object):
   def __init__(self,autor,nota):
      self.id=None
      self.autor=autor
      self.nota=nota
   def __str__(self):
      return self.nota

class NotaApp():
   _cp_config={'tools.sessions.on':True}


   def _render_nota(self,nota):
      cherrypy.HTTPRedirect('/')
      return vista_nota %(nota,nota.autor)



   @cherrypy.expose
   def index(self):
      autor=cherrypy.session.get('autor',None)
      page=[cabecalho]
      if autor:
```

```
            page.append("""Oi %s deixe sua nota."""%(autor,))
            page.append("""<a href="autor"> Mudando Autor.</a>""")
            page.append(form_nota)
        else:
            page.append("""<a href="autor">Coloque seu nome.</a>""")
        nota=notas[:]
        for nota in notas:
            page.append(self._render_nota(nota))
        page.append(rodape)
        cherrypy.HTTPRedirect('/')
        return page

    @cherrypy.expose
    def post(self,text):
        autor=cherrypy.session.get('autor',None)
        nota=Nota(autor,text)
        notas.append(nota)
        raise cherrypy.HTTPRedirect('/')


if __name__=='__main__':

    nota=NotaApp()
    nota.autor=Autor()
    cherrypy.tree.mount(nota,'/')
    cherrypy.engine.start()
```

8. O Servidor HTTP interno - o Cherrypy tem seu próprio servidor web. Este servidor web é a porta para uma aplicação Cherrypy através do qual atendem-se todas as requisições e respostas HTTP. Pode-se usar outro servidor se necessário. Internamente temos o motor Cherrypy responsável pela criação e gerenciamento de objetos Request e Response, para iniciar este motor deve-se emitir o comando **cherrypy.engine.start**.

8.1 Configuração – para parametrizar o servidor Cherrypy HTTP e seu tratamento de pedido de URI podemos armazenar suas configurações em um arquivo texto com formato INI ou em um dicionário Python. Estas configurações são passadas para o servidor através da instância cherrypy.config.update() e por pedido através do método cherrypy.tree.mount.
Exemplo: Dicionário

```
global_conf = {
        'global': {
                'server.socket_host': 'localhost',
                'server.socket_port': 8080,
                },
            }
application_conf = {
                '/style.css': {
                        'tools.staticfile.on': True,
                        'tools.staticfile.filename': os.path.join(_curdir,
                                                'style.css'),
                        }
```

```
        }

Arquivo: use sempre objetos válidos em Python (string, inteiro,booleano...)

[global]
server.socket_host="localhost"
server.socket_port=8080

[/style.css]
tools.staticfile.on=True
tools.staticfile.filename="/full/path/to.style.css"
```

Observações:

Note que quando indicamos um arquivo de estilo **/style.css** devemos indicar seu endereço absoluto uma vez que ele será manipulado por uma ferramenta chamada **staticfile**. Salientamos que para o CherryPy efetuar estas configurações ele precisa chamar cherrypy.config.update(global_conf) para o dicionário e cherrypy.config.update('path_para_o_config/arquivo') para a configuração no arquivo.

Nós também devemos passar os valores de configuração para a montagem da aplicação. Com o dicionário chamamos cherrypy.tree.mount(instancia_da_aplicação, nome_script, config=application_conf), e com o arquivo cherrypy.tree.mount(instancia_da_aplicação, nome_script, config=application_conf),

Outra forma de configuração de sua aplicação é o a atributo _cp_config no seu manipulador de página ou o atributo de classe contendo os manipuladores que atuará sobre todos os manipuladores.

Exemplo com a configuração dos manipuladores:

```
import cherrypy
class Root:
    _cp_config = {'tools.gzip.on': True}

    @cherrypy.expose
    def index(self):
        return "Bem vindo"

    @cherrypy.expose
    def default(self, *args, **kwargs):
        return "Pagina inexistente"

    @cherrypy.expose
    # a linha abaixo eh menos usada, mas eh uma forma de usar um decorador
    # mas devemos lembrar que temos que setar o atributo da classe _cp_config
    @cherrypy.tools.gzip()
    def echo(self, msg):
        return msg

    @cherrypy.expose
    def alo(self):
```

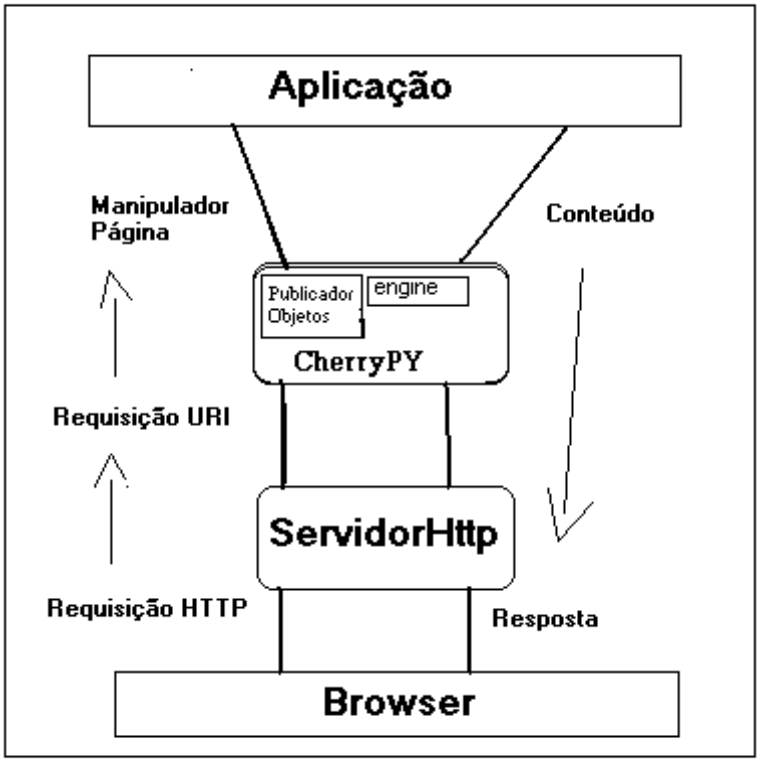
```
        return "Ola"
    # desabilitamos o gzip da pagina alo
    alo._cp_config = {'tools.gzip.on': False}

if __name__ == '__main__':
    cherrypy.quickstart(Root(),'/')
    # vai executar em http://localhost:8080
    # a montagem da aplicacao em um local diferente de '/' por exemplo '/local'
    # nao deve ser colocada no arquivo de setting por que ele eh da aplicacao
    # e nao da montagem, mudando-se a montagem muda-se apenas o local onde acessamos
    # a pagina no browser ou seja http://localhost:8080/local/
```

9. Publicando e Expondo Objetos no CherrPy – diferente do **lighttpd** e do **Apache** que mapeiam a Requisições de URI no sistema de arquivos(eficiente para websites estáticos), o CherryPy usa um algoritmo próprio interno para recuperar o manipulador da Requisição URI. A versão CherryPy 2.0 passa a usar uma árvore de objetos Python publicáveis, com isto surge o conceito Publicado e o conceito Exposto. Um objeto é dito publicado quando ele é agregado a uma árvore de objetos e a raiz dela é montada no CherryPy através da chamada cherrypy.tree.mount, e é dito Exposto quando o atributo expose do objeto for posicionado para True. Um objeto deve ser então publicado e exposto para manipular uma URI no servidor CherryPY.
Exemplo:

```
#objeto publicado root e o objeto admin que é um atributo do objeto root.
root=Blog()
root.admin=Admin()
cherrypy.tree.mount(root,'/blog')

#aqui mostramos como deixar o objeto visível para que ele seja uma resposta URI
# note que se requisitarmos /coisa retornará um erro 404 Not Found a razão é que o objeto
#é publicado e não exposto, um objeto exposto é chamado page handler(manipulador de página
# pela comunidade)
import cherrypy
class Root:
      @ cherrypy.expose
       def index(self):
            return self.coisa()
        # objeto publicado mas não exposto
       def coisa(self):
            return “alguma coisa”
cherrypy.tree.mount(Root(),'/')
cherrypy.engine.start()
# execute os comandos: http://localhost:8080/ e http://localhost:8080/coisa
```

9.1 Métodos index() e default() - são métodos especiais de manipulação de páginas o index trata a página “/” ou página raiz e o default é usado quando o manipulador da página não é encontrado e assim permite tratar páginas irregulares, parâmetros também podem ser passados pela URI. No exemplo abaixo damos um exemplo onde usamos parâmetros posicionais para passagem de dados, desta forma ao passarmos a URI http://localhost:8080/alo/sergio ou http://localhost:8080/alo?x=sergio note que sergio é um parâmetro da URI.

```
import cherrypy
class Root:
    _cp_config = {'tools.gzip.on': True}

    @cherrypy.expose
    def index(self):
        return "Bem vindo"

    @cherrypy.expose
    def default(self, *args, **kwargs):
        return "Pagina inexistente"

    @cherrypy.expose
    # a linha abaixo eh menos usada, mas eh uma forma de usar um decorador
    # mas devemos lembrar que temos que setar o atributo da classe _cp_config
    @cherrypy.tools.gzip()
    def echo(self, msg):
        return msg

    @cherrypy.expose
    def alo(self,x=""):
        return "Ola ",x
    # desabilitamos o gzip da pagina alo
    alo._cp_config = {'tools.gzip.on': False}

if __name__ == '__main__':
    cherrypy.quickstart(Root(),'/')
    # vai executar em http://localhost:8080
    # a montagem da aplicacao em um local diferente de '/' por exemplo '/local'
    # nao deve ser colocada no arquivo de setting por que ele eh da aplicacao
    # e nao da montagem, mudando-se a montagem muda-se apenas o local onde acessamos
    # a pagina no browser ou seja http://localhost:8080/local/
```



10. Módulos da biblioteca CherryPY – um conjunto de módulos compõe a biblioteca cherrypy estes módulos fazem tarefas como: gerenciamento de sessão, serviço de recurso estático, manipulador de codificação e armazenamento de dados básicos(cache).

10.1 Característica de AutoCarga – o CherryPy é um longo processo de execução Python, isto significa que podemos modificar o módulo de aplicação sem propagar para o processo existente. A tarefa de reinicializar o servidor é tediosa e por isso temos uma forma de reinicialização chamada Autoreload, que executa o servidor com o novo módulo assim que ele é modificado. Este autoreload pode ser configurado através do arquivo de configuração, ele na realidade checa de tempos em tempos a modificação e então executa o reload. O valor default é 1s.

```
[global]
server.environment = "production"
engine.autoreload_on = True
engine.autoreload_frequency = 5
```

10.2 Módulo Caching – o caching é um lado importante de uma aplicação web porque ele reduz a carga e o stress de diferentes servidores em ação (HTTP, Banco de Dados e Aplicação). O módulo do CherryPy trabalha com o servidor de HTTP armazenando as informações geradas na saída e as fornecidas pelo usuário.

10.3 Módulo de Análise(coverage) – é muito benéfico sabermos qual o rumo tomado por uma aplicação com entrada de dados, uma vez que isto nos ajuda a determinar possíveis gargalos e como a aplicação se comporta. Este módulo aqui chamado de análise fornece uma saída amigável através do browser (este é um dos poucos módulos que possui pacotes internos de terceiros).

10.4 Módulo de Codificação e Decodificação - publicar na web significa lidar com a

multiplicidade de codificação de caracteres existentes. Por um lado você só pode publicar seu próprio conteúdo usando o US-ASCII sem pedir feedback dos leitores e por outro lado você pode liberar um aplicativo com um boletim que irá lidar com qualquer tipo de conjunto de caracter. Para ajudar nesta tarefa CherryPy fornece um módulo de codificação / decodificação que filtra o conteúdo de entrada e saída com base em servidor ou configurações do usuário.

10.5 Módulo HTTP – este módulo oferece um conjunto de classes e funções para manipular cabeçalho e entidades HTTP.
Por exemplo:

```
s='GET /nota/1 HTTP/1.1'
r=http.parse_request_line(s) # r conterá a lista('GET', '/nota/1, ' ','HTTP/1.1')
s='GET /nota?=id=1 HTTP/1.1'   # string com parâmetro id=1
r=http.parse_request_line(s) #r conterá a lista ('GET', '/nota', 'id=1','HTTP/1.1')
http.parseQueryString(r[2]) # retorna {'id':'1'}
```

Para fornecer uma clara interface para o cabeçalho HTTP veja exemplo:

```
accept_value =
text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,
*/*;q=0.5"
values = http.header_elements('accept', accept_value)
print values[0].value, values[0].qvalue # irá imprimir text/html 1.0
```

10.6 Módulo Httpath – o módulo http/1.0 possui uma especificação de um esquema de autenticação de acesso básico que é inseguro (chamado BASIC) e um esquema de autenticação criptografado seguro (chamado DIGEST), este módulo prevê esta implementação de acordo com a RFC 2617.

10.7 Módulo Profiler – este módulo é uma interface para checar a performance da aplicação.

10.8 Módulo Sessão – construída em cima de um protocolo sem estado, a web não armazena informações e isto significa que cada pedido ou comunicação é independente. Apesar desta situação quando navegamos em um site de comércio eletrônico temos a sensação de que a cada comunicação a web está armazenando as informações, isto não é verdade o que ocorre é que foi implantado um mecanismo chamado de sessão que permite aos servidores controlar estas informações. No CherryPy o módulo sessão é que implementa esta interface de um modo simples permitindo ao aplicativo armazenar, recuperar, alterar e excluir dados de um objeto chamado sessão. São três os objetos de armazenamento que fazem este tratamento:

RAM – armazenamento eficiente, aceita qualquer tipo de informação e não precisa ser configurado. Por trás disto temos uma informação que pode ser perdida quando o servidor sai doa r e o consumo de memória pode crescer rapidamente.

File System – persistência da informação, configuração simples, mas ele tem deficiências quanto ao travamento de arquivos e somente pode armazenar objetos de modo serial(módulo pickle do Python).

Relational Database – persistência da informação, robustez, escalabilidade e balanceamento de carga, tem como deficiências configuração mais difícil.

Observação: sua aplicação usa uma interface de alto nível que independe do modelo de sessão

usado e que pode ser modificado ser mexer na sua aplicação.

10.9 Módulo Estático – mesmo as aplicações dinâmicas possuem recursos estáticos como imagens e arquivos CSS. O CherryPy fornece um módulo capaz de atender essas necessidades ou servir até uma estrutura de diretórios completa, podendo até mesmo lidar com o uso de cabeçalho HTTP do tipo If-Modified-Since que verifica se um recurso foi modificado desde uma determinada data, evitando processamento desnecessário.

10.10 Módulo Tidy – voltado para checar se o código gerado pela aplicação é válido e claro, o CherryPy fornece este módulo capaz de facilmente filtrar o conteúdo de saída com o uso de ferramentas como tidy ou nsgml.

10.11 Módulo Wsgiapp – este módulo permite que qualquer aplicação WSGI use o CheryyPy.

10.12 Módulo XML-RPC – este módulo permite transformar um objeto publicado em um serviço XML-RPC. O CherryPy irá extrair a partir do documento XML de entrada, o nome do método, os valores e aplicar a mesma lógica como se fosse uma chamada URI qualquer, ou seja procurar seu manipulador de página correspondente. O manipulador de página então retorna o conteúdo de resposta dentro de um XML-RPC válido e envia de volta ao cliente.
Exemplo:

Servidor CherryPy:

```
import cherrypy
from cherrypy import _cptools
class Root(_cptools.XMLRPCController):
    @cherrypy.expose
    def echo(self, message):
        print message  # para visualizar no servidor a mensagem recebida que será echoada
        return message

if __name__ == '__main__':
        cherrypy.quickstart(Root(), '/xmlrpc')
```

Seu cliente poderia ser assim:

```
import xmlrpclib
proxy = xmlrpclib.ServerProxy('http://localhost:8080/xmlrpc/')
proxy.echo('Alo') # retornara alo
```

Para executar este programa e visualizar o resultado entre no modo DOS e execute o servidor.
Após isto execute no shell (IDLE) o programa cliente e você verá que o servidor recebeu um dado XML e imprimiu a mensagem.

10.13 Tools – o CherryPy fornece uma interface unificada referenciada como tool para chamar estes módulos e chamar seus próprios módulos. Um TOOL é uma interface que estende o CherryPy permitindo conectá-lo a componentes de terceiros. Para configurar TOOLS temos 3 formas:
Arquivo ou Dicionário:

```
conf={'/':{
                'tools.encode.on': True,
                ´tools.encode.encoding':'ISO-8859-1'
```

```
            }
        }
    cherrypy.tree.mount(Root(),'/',config=conf)
```

Agregado a um manipulador de página:

```
    @cherrypy.expose
    @cherrypy.tools.encode(encoding='ISO 8859-1')
    def index(self)
        return 'Olá'
```

Fazendo uma chamada a biblioteca com interface de alto nível:

```
    def index(self):
        cherrypy.tools.accept.callable(media='text/html')
        # aceitando o tipo media dentro da cabeçalho Accept do HTTP
```

10.14 Manipulando Erros e Exceções: o CherryPy vai tratar erros e exceções como uma aplicação Python, pegando esses erros e exceções e transformando-os em mensagens HTTP quando necessário. Um erro HTTP 500 é retornado quando uma exceção ocorre e não é tratada por outras partes do pedido. A especificação do HTTP define dois conjuntos de códigos de erros, 4XX erros do cliente e 5XX erros do servidor. Erro 4XX ocorre quando o usuário envia uma requisição inválida e erro do servidor informa ao usuário que um evento ocorreu e que o servidor não executará integralmente o processo de requisição.

```
import cherrypy
class Root:
   @cherrypy.expose
   def index(self):
      raise NotImplementedError, "Isto eh um erro..."
if __name__ == '__main__':
cherrypy.quickstart(Root(), '/')
```

Veja a mensagem:

500 Internal Server Error

The server encountered an unexpected condition which prevented it from fulfilling the request.

```
Traceback (most recent call last):
  File "C:\Python25\lib\site-packages\cherrypy\_cprequest.py", line 606, in respond
    cherrypy.response.body = self.handler()
  File "C:\Python25\lib\site-packages\cherrypy\_cpdispatch.py", line 25, in __call__
    return self.callable(*self.args, **self.kwargs)
  File "C:/Python25/cherry/erro500.py", line 5, in index
    raise NotImplementedError, "Isto eh um erro..."
NotImplementedError: Isto eh um erro..
```

Powered by CherryPy 3.1.2

Para desabilitar este modo de acompanhamento do erro (traceback) podemos usar o comando request.show_tracebacks na seção global.

```
# exemplo do modo traceback
import cherrypy
class Root:
    @cherrypy.expose
    def index(self):
        raise NotImplementedError, "Isto é um erro..."
if __name__ == '__main__':
    global_conf={
        'global':{'request.show_tracebacks':False
                  }}
    application_conf={
        '/':{
              'tools.encode.encoding':'ISO-8859-1',
              'tools.encode.on':'True'
            }}

    cherrypy.config.update(global_conf)
    cherrypy.quickstart(Root(), '/',config=application_conf)
```

Retorno do erro:

500 Internal Server Error

The server encountered an unexpected condition which prevented it from fulfilling the request.

Powered by CherryPy 3.1.2

10.14.1 Interface de tratamento de erros – uma interface simples é fornecida ao usuário permitindo ao desenvolvedor enviar seu código de erro.

Cherrypy.HTTPError(codigo_de_erro,[mensagem_de_erro])

```
import cherrypy
class Root:
   @cherrypy.expose
   def index(self):
      raise cherrypy.HTTPError(401, 'Voce nao pode acessar este \
recurso')

if __name__ == '__main__':
   cherrypy.quickstart(Root(), '/')
```

mensagem:

401 Unauthorized

Voce nao pode acessar este recurso

```
Traceback (most recent call last):
  File "C:\Python25\lib\site-packages\cherrypy\_cprequest.py", line 606, in
respond
    cherrypy.response.body = self.handler()
  File "C:\Python25\lib\site-packages\cherrypy\_cpdispatch.py", line 25, in
__call__
    return self.callable(*self.args, **self.kwargs)
  File "interfaceerro.py", line 6, in index
    recurso')
HTTPError: (401, 'Voce nao pode acessar este recurso')
```

Powered by CherryPy 3.1.2

Outro exemplo sem o traceback:

```
import cherrypy
class Root:
   @cherrypy.expose
   def index(self):
   # shortcut to cherrypy.HTTPError(404)
      raise cherrypy.HTTPError(404,'Caminho / nao encontrado')
if __name__ == '__main__':
   conf = {'global':{'request.show_tracebacks':False}}
   cherrypy.config.update(conf)
   cherrypy.quickstart(Root(), '/')
```

Resposta:

404 Not Found

Caminho / nao encontrado

Powered by CherryPy 3.1.2

Não se esqueça para executar o servidor basta digitar: http://localhost:8080/ Nogueira

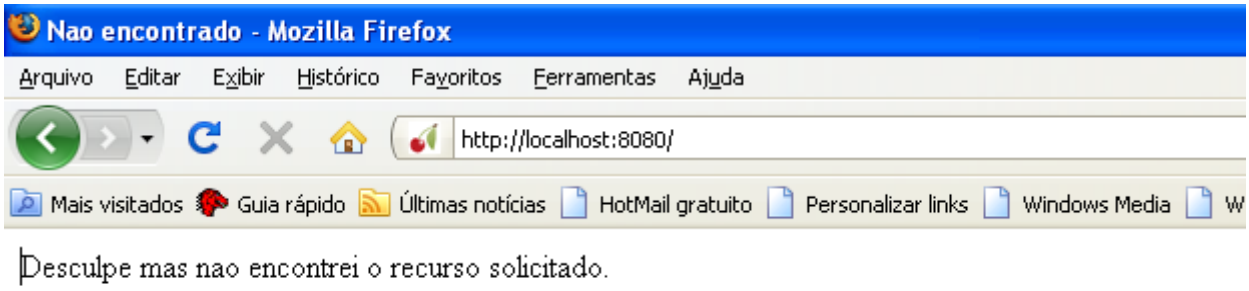
10.14.2 Interface de Erro modificada por um arquivo .html – podemos modificar a saída de erro fornecendo um arquivo html que responda ao erro.

```
import cherrypy
class Root:
    @cherrypy.expose
    def index(self):
        raise cherrypy.NotFound
if __name__ == '__main__':

    cherrypy.config.update({'global':{'error_page.404':
'notfound.html' }})
    cherrypy.quickstart(Root(), '/')
```

arquivo html: notfound.html

```
<html>
    <head><title>Nao encontrado</title></head>
    <body>
        <p>Desculpe mas nao encontrei o recurso solicitado.</p>
    </body>
</html>
```

resultado:



11. O CherryPy e o protocolo HTTP – evoluindo lentamente o CherryPY primeiramente implementou a versão do HTTP/1.0 e depois implementou a HTTP/1.1. Ele é compatível condicionalmente com o HTTP/1.1 e implementa todos os níveis que deve e será requerido, mas não todos os níveis que deveriam ser da especificação. Ele suporta as seguintes características:

- Para apoiar o HTTP/1.1 devemos enviar um campo de cabeçalho do Host em qualquer requisição efetuada, senão o CherryPy emitirá um código de erro 400 (secção 14.23 do RFC 2616)
- CherryPy gera um campo de cabeçalho de data em todas as configurações(14.18 do RFC 2616)
- CherryPy manipula a resposta Continue código 100 em clientes que o suportam.
- O servidor HTTP CherryPy suporta conexões persistentes que são o padrão em HTTP/1.1,

através da utilização da Conexão: cabeçalho Keep-Alive. Esteja ciente de que a mudança do servidor HTTP pode quebrar essa compatibilidade, se o servidor escolhido não suporta tal recurso.

- O CherryPy manipula corretamente respostas e requisições fragmentadas
- O CherryPy suporta e responde corretamente requisições do tipo If-Modified-Since(se modificada desde) ou If-Unmodified-Since.
- Pode-se utilizar qualquer método HTTP.
- CherryPy lida com todas as combinações de versões HTTP entre o cliente e a configuração definida para o servidor.
- O protocolo do servidor pode ser modificado através server.protocol_version.

12. Múltiplos Servidores HTTP – por definição sempre que iniciamos o CherryPy, ele é iniciado numa simples instância de servidor HTTP. O CherryPy ainda fornece uma API para rodar diferentes instâncias de servidores HTTP num único processo.

Inicialização do CherryPy HTTP em sua própria Thread(modo usual):

```
conf = {'global': {'server.socket_port': 100100,
                   'server.socket_host': 'localhost'}}
cherrypy.config.update(conf)
cherrypy.server.quickstart()
```

Aplicação do CherryPy que pode atender diferentes interfaces de redes:

```
from cherrypy import _cpwsgi

# Criando um servidor na interface 1102.168.0.12 porta 100100
s1 = _cpwsgi.CPWSGIServer()
s1.bind_addr = ('1102.168.0.12', 100100)

# Criando um servidor na interface 1102.168.0.27 porta 4700
s2 = _cpwsgi.CPWSGIServer()
s2.bind_addr = ('1102.168.0.27', 4700)

# Informando ao CherryPy qual servidor iniciar e usar
cherrypy.server.httpservers = {s1: ('1102.168.0.12', 100100),
                               s2: ('1102.168.0.27', 4700)}
cherrypy.server.start()
```

No exemplo anterior criamos duas instâncias do servidor HTTP e para cada uma nós definimos uma endereço de ligação na qual o socket ouvirá os pedidos requisitados. Depois agregamos os servidores HTTP e chamamos o método start. Veja que não chamamos cherrypy.config.update que é utilizado para configurações globais de todos servidores. Mas podemos utilizar a configuração global e ainda posicionar cada instância. Veja a seguir:

```
s1.socket_port = 100100
s1.socket_host = '1102.168.0.12'
```

```
s1.socket_file = ''
s1.socket_queue_size = 5
s1.socket_timeout = 10
s1.protocol_version = 'HTTP/1.1'
s1.reverse_dns = False
s1.thread_pool = 10
s1.max_request_header_size = 500 * 1024
s1.max_request_body_size = 100 * 1024 * 1024
s1.ssl_certificate = None
s1.ssl_private_key = None
```

13. Servidor de Aplicação Multi-Threaded (Múltipla Linha de Execução) – embora aparentemente não pareça mas a aplicação (cherrypy.request ou cherrypy.response)é executada em ambiente multi-thread. Devemos também saber que cherrypy.request e cherrypy.response são estrutura de dados do tipo thread o que implica que sua aplicação pode chamá-las independentemente. Como padrão todas as vezes que chamamos um servidor CherryPy um conjunto de threads é criado para tratar os pedidos, o número de threads padrão é 10 e pode ser configurado através de server.thread_pool. Para cada aplicação devemos determinar o melhor número através de testes.

14. Escalonador de URI – além de mapear no Python as URI um atributo chamado exposed deve ser setado para exibir a página. Com o passar do tempo a comunidade necessitava de outros métodos mais flexíveis e de outros escalonadores e foi prontamente atendida pelo CherryPy 3 que possui outros 3 tipos.

14.1 Escalonador de Método HTTP – fornece URIs independentes das ações a serem executadas pelo servidor de recurso. Exemplo:http://meuhost.com/album/delete/12
Veja que temos na URI a operação que desejamos efetuar, que seria mapeada no CherryPY da seguinte forma: album.delete(12)
Outra forma seria remover a operação da URI e ficaria assim: http://meuhost.com/album/12
e a informação é enviada pela requisição do HTTP da seguinte forma: DELETE /album/12 HTTP/1.1

Exemplo do escalonador de URI no CherryPy:

```
class Album:
    exposed = True
    def GET(self, id):
        ....
    def POST(self, title, description):
        ....
    def PUT(self, id, title, description):
        ....
    def DELETE(self, id):
        ....
```

Quando usamos o escalonador de métodos HTTP, o manipulador de páginas chamado será album.DELETE(12), no caso do exemplo. E temos a classe toda exposta através da linha exposed=True. Na realidade quando usamos o escalonador de método HTTP, o manipulador é de fato uma representação conceitual do recurso direcionado pela URI. O escalonador verifica se a classe tem um método correspondente ao nome do método HTTP usado para a sua solicitação.

Caso não possua um código de erro 405 é enviado para informar que não é possível executar a operação. Este método HTTP pode ser utilizado setando request.dispatch.
{'/':{'request.dispatch':cherry.dispatch.MethodDispatcher()}}

14.2 Escalonador de Rotas – é função do motor CherryPy encontrar a melhor correspondência seja usando o método de objetos ou o escalonador de métodos HTTP, mas alguns desenvolvedores preferem uma abordagem mais explícita e preferem mapear explicitamente as URIs. Com este método você deve conectar um padrão que corresponda a URI e especificar um manipulador de página. Para utilizar o escalonador de rotas precisamos instalar o módulo routes (import routes), dentro dele temos um método chamado connect que fará o escalonamento das rotas.
connect(name, route,controller,**kwargs)
name – é o único nome para conectar a rota
route – é o padrão para encontrar a URI
controller – instância contendo o manipulador de página
**kwargs – parâmetros extras válidos para uma rota

```
import cherrypy
import routes
class Root:
    def index(self):
        return "Retornou."
    def alo(self, name):
        return "Alo %s" % name
if __name__ == '__main__':
    root = Root()

# Cria instância do escalonador
    d = cherrypy.dispatch.RoutesDispatcher()

# conecta uma rota que será manipulada pelo index
    d.connect('default_route','', controller=root)
# conecta uma rota para o alo
# isto encontrará a URI como /diga/alo/jose'
# mas nao alo/jose
    d.connect('diga','diga/:action/:name',controller=root,action='alo')
# seta escalonador
    conf = {'/': {'request.dispatch': d}}
    cherrypy.quickstart(root, '/', config=conf)
```

Obs: é necessário instalar o pacote Routes com o EasyInstall (setuptools-0.6c11.win32-py2.5.exe (http://pypi.python.org/packages/2.5/s/setuptools/setuptools-0.6c11.win32-py2.5.exe) – instala o EasyInstall para o Python 2.5 e Routes-1.11-py2.6.egg (http://pypi.python.org/packages/2.6/R/Routes/Routes-1.11-py2.6) é o arquivo que devemos instalar com o EasyInstall).

14.3 Escalador de Servidor Virtual – pode acontecer que você necessite dentro do servidor cherry de uma aplicação diferente para cada domínio. Com o CherryPy isto é possível veja o exemplo abaixo:

```
import cherrypy
```

```
class Site:
    def index(self):
        return "Alo, mundo"
    index.exposed = True

class Forum:
    def __init__(self, name):
        self.name = name

    def index(self):
        return "Benvindo ao forum %s " % self.name
    index.exposed = True

if __name__ == '__main__':
    site = Site()
    site.carros = Forum('Carros')
    site.musicas = Forum('Musicas')
    hostmap = {'www.carros.com': '/carros',
        'www.musicas.com': '/musicas',}
    cherrypy.config.update({'server.socket_port': 80})
    conf = {'/': {'request.dispatch':
                cherrypy.dispatch.VirtualHost(**hostmap)}}
    cherrypy.tree.mount(site, config=conf)
    cherrypy.engine.start()
```

Para testar o exemplo altere o arquivo HOSTS da sua máquina e adicione as seguintes linhas no final do arquivo:

127.0.0.1 www.musicas.com
127.0.0.1 www.carros.com

Automaticamente, quando receber estes domínios, o browser direcionará para o servidor local e o cherrypy atenderá a requisição de forma correta.

14. Hook(desvio para uma rotina de futuros processos) no núcleo da máquina CherryPy – um dos mais potentes aspectos do CherryPy é como seu núcleo permite que seu comportamento normal seja modificado com uma fina granularidade, esse mecanismo é chamado HOOKING para customização do núcleo. Um ponto de entrada pode ser chamado durante o processo de requisição, o CherryPy fornece os seguintes pontos: **on_start_resource** – inicio do processo, **before_request_body** – antes de o CherryPy tentar ler o corpo da requisição, **before_handler** – antes do manipulador de página, **before_finalize** – antes de iniciar o processo de resposta, depois da chamada ou não do manipulador de páginas, **on_end_resource** – chamado quando o processamento do recurso terminar, **before_error_response** e **after_error_response** – permite a recuperação do erro ou tomar outra decisão e **on_end_request** - antes de terminar o link com o cliente.

Para estabelecer um ponto de entrada digite o comando:

cherrypy.request.hooks.attach(ponto, chamada, failsafe=None,priority=None, **kwargs)

ponto – define qual ponto de entrada on_start_resource,....., visto anteriormente.

chamada – qual método atende a esta chamada

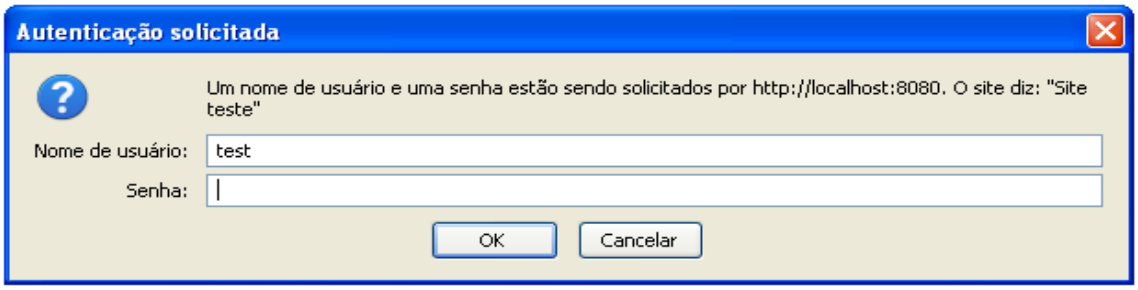
failsafe – mesmo que outra chamada falhar, esta chamada deve ser executada. É um parâmetro útil pois permite o sistema ser flexível e recuperar-se de problemas quando eles ocorrerem.

priority- de 0 a 100 ordena as chamadas, valores menores maiores prioridades.

Você pode anexar várias chamadas, ao ponto de chamada, mas quanto mais chamadas mais lento o sistema fica naquele ponto. Este mecanismo é muito usado em chamadas de alto nível denominadas de ferramentas(tools) e não diretamente.

15. Caixa de Ferramentas do CherryPy – enquanto refatorava o CherryPy Robert Brewer projetou uma interface de alto nível (tool). O objetivo era oferecer ferramentas pronta-pra-uso capaz de realizar tarefas comuns com uma API flexível e amigável. As ferramentas construídas dentro do CherryPy oferecem uma interface simples para chamada da biblioteca. Elas são usadas de 3 formas diferentes: para configuração, como decorador ou via atributo _cp_config de uma página e como uma chamada Python que pode ser aplicada de dentro de qualquer função.

15.1 Ferramenta Básica de Autenticação – fornece a autenticação básica para sua aplicação(RFC 2617). Possui os seguintes argumentos: realm (fornecido pelo desenvolvedor ela aparece no cabeçalho da caixa de autenticação), users(dicionário da forma usuário:senha) e encrypt (default é None(método hash MD5) serve para acionar uma função para criptografar).




```
import sha
import cherrypy
class Root:
    @cherrypy.expose
    def index(self):
        return """<html>
                  <head></head>
                    <body>
                      <a href="admin">Admin area</a>
                    </body>
                </html>
"""
class Admin:
    @cherrypy.expose
    def index(self):
        return "Area administrativa"

if __name__ == '__main__':
    def get_users():
        # 'test': 'test'
        return {'test':'a94a8fe5ccb19ba61c4c0873d391e987982fbbd3'}
    def encrypt_pwd(token):
        print sha.new(token).hexdigest()
        return sha.new(token).hexdigest()
```

```
conf = {'/admin': {'tools.basic_auth.on': True,
        'tools.basic_auth.realm': 'Site teste',
        'tools.basic_auth.users': get_users,
        'tools.basic_auth.encrypt': encrypt_pwd}}
root = Root()
root.admin = Admin()
cherrypy.quickstart(root, '/', config=conf)
```

Para testar o sistema digite: Usuário:test e Senha: test

Veja a função retorna o dicionário usuário:senha que pode estar localizado num banco de dados. O esquema de autenticação básica não é totalmente seguro já que a senha codificada pode ser decodificada durante a transmissão. Mas é usado sobre o SSL que encripta os dados.

15.2 Ferramenta de Caching(armazenamento) – A proposta desta ferramenta é fornecer um armazenamento na memória do conteúdo gerado pelo cherrypy. Os argumentos são invalid_methods(tuplas que armazenam strings de métodos HTTP como Post, Pust, Delete que não podem ser amazenados) e cache_class(classe de objeto a ser usada para armazenamento).

15.3 Ferramenta de codificação – esta ferramenta serve para codificar o conteúdo de resposta num código definido. Os argumentos são encoding (None – verifica o header Content-Type e tenta localizar o conjunto de caracteres se não encontrar usa o default) e errors(define como a ferramenta deve reagir quando falha a codificação do caracter).

```
import cherrypy
from cherrypy import tools
class Root:
   @cherrypy.expose
   def index(self):
      return """<html>
                <head></head>
                <body>
                   <form action="alo" method="post">
                      <input type="text" name="name" value="" />
                   </form>
                </body>
             </html>"""

   @cherrypy.expose
   @tools.decode(encoding='iso-8859-1')
   def alo(self, name):
      return "Alo %s" % (name, )

if __name__ == '__main__':
   cherrypy.quickstart(Root(), '/')
```

15.4 Ferramenta de Autenticação Digest – A proposta é fornecer a autenticação definida na norma RFC 2617. Os argumentos são realm (domínio-o que você tá acessando) e users (dicionário

usuário:senha, ou uma chamada que retorna este dicionário). Esta ferramenta não fornece uma maneira de enviar uma senha criptografada. A razão para isto é que o esquema digest não envia a senha através da rede claramente. A forma que ele envia é a seguinte: primeiro o cliente requisita acesso ao recurso e o servidor retorna o erro 401 indicando o esquema digest e um aviso para esta transação. Após receber este aviso uma nova mensagem é gerada com este aviso, a senha e o usuário, isto é gerado através do algoritmo MD5. Por último após receber esta nova mensagem o servidor tenta gerar o mesmo valor e então conseguindo a autenticação é permitida.

```
import cherrypy

class Root:
    @cherrypy.expose
    def index(self):
        return """<html>
<head></head>
<body>
<a href="admin">Area administrativa</a>
</body>
</html>
"""

class Admin:
    @cherrypy.expose
    def index(self):
        return "Area Privada"

if __name__ == '__main__':
    def get_users():
        return {'test': 'test'}
    conf = {'/admin': {'tools.digest_auth.on': True,
            'tools.digest_auth.realm': 'Site do Sergio',
            'tools.digest_auth.users': get_users}}
    root = Root()
    root.admin = Admin()
    cherrypy.quickstart(root, '/', config=conf)
```

Utilize usuário = test e senha= test para acessar a área privada.

15.5 Ferramenta de Redirecionamento de Erro – a proposta desta ferramenta é modificar o manipulador de erro do Cherrypy. Os argumentos são url(a url para onde você quer direcionar default='') e internal(True – o redirecionamento é escondido do cliente ocorrendo só dentro do contexto da requisição e False – o cherrypy informa ao cliente que o redirecionamento deve ser fornecido pelo cliente para a URL dada).

15.6 Ferramenta Etag – tem como propósito validar uma entidade com um marcador enviado pelo usuário e a resposta é gerada de acordo com o definido pela RFC 2616 seção 14.24. Etags é uma forma de o HTTP armazenar respostas e então diminuir a sobrecarga das partes envolvidas.
Na primeira requisição do manipulador de página index, a ferramenta gerará um valor etag e insere ele no cabeçalho de resposta. Na próxima requisição da URI, o cliente incluirá a última etag

recebida. A ferramenta compara com a corrente e se encontrar a resposta será 304 Not Modified informando que o cliente pode seguramente usar a cópia do recurso.
Veja que se você necessita que o valor de etag seja computado por diferentes razões, a melhor forma é setar o parâmetro autotags para False, o default, e então adicione através de seu manipulador de página seu Etag no cabeçalho de resposta.

```
import cherrypy
from cherrypy import tools

class Root:
    @cherrypy.expose
    def index(self):
        return """<html>
<head></head>
<body>
<form action="hello" method="post">
<input type="text" name="name" value="" />
</form>
</body>
</html>
"""

    @cherrypy.expose
    def hello(self, name):
        return "Hello %s" % name

if __name__ == '__main__':
    conf = {'/': {'tools.etags.on': True,
            'tools.etags.autotags': True}}
    cherrypy.quickstart(Root(), '/', config=conf)
```

15.9 Ferramenta Gzip – tem como proposta compactar o contéudo codificado no corpo de resposta. Os argumentos são compress_level (nível de compressão default=10) e mime_types(['text/html','text/plain'] lista de tipos comprimíveis).

```
import cherrypy
from cherrypy import tools

class Root:
    @cherrypy.expose
    @tools.gzip()
    def index(self):
        return "Isto foi comprimido"

if __name__ == '__main__':
    cherrypy.quickstart(Root(), '/')
```

O gzip não deve ser usado quando a resposta é uma stream.

15.10 Ferramenta para Ignorar Cabeçalho – a proposta desta ferramenta é remover o cabeçalho específico do HTTP antes de eles serem processados pelo Cherrypy. O argumento é ignore_headers

que é uma tupla de nomes(default = > headers=('Range',)).

```
import cherrypy
from cherrypy import tools

class Root:
    @cherrypy.expose
    @tools.ignore_headers(headers=('Accept-Language',))
    def index(self):
        return "Accept-Language: %s" \
            % cherrypy.request.headers.get('Accept-Language','nao fornecida')

    @cherrypy.expose
    def outra(self):
        return "Accept-Language: %s" % cherrypy.request.headers.get('Accept-Language')

if __name__ == '__main__':
    cherrypy.quickstart(Root(), '/')

Digite:http://localhost:8080 e http://localhost:8080/outra para visualizar a execução.
```

15.11 Ferramenta para Logar Cabeçalho – tem como objetivo mostrar os cabeçalhos de requisição dentro de um arquivo de log quando um erro ocorrer no servidor. Por default ela é desabilitada.

```
import cherrypy
from cherrypy import tools
class Root:
    @cherrypy.expose
    def index(self):
        raise StandardError, "Some sensible error message here"
if __name__ == '__main__':
    cherrypy.config.update({'global': {'tools.log_headers.on':
                        True}})
    cherrypy.quickstart(Root(), '/')
```

15.12 Ferramenta para logar caminho do erro – tem como objetivo mostrar o caminho do erro num log quando o mesmo ocorrer. Ela é habilitada por default.

```
import cherrypy
from cherrypy import tools
class Root:
@cherrypy.expose
def index(self):
raise StandardError, "Aqui mensagem de erro"
if __name__ == '__main__':
# This tool is applied globally to the CherryPy process
# by using the global cherrypy.config.update method.
```

```
cherrypy.config.update({'global': {'tools.log_tracebacks.on':
False}})
cherrypy.quickstart(Root(), '/')
```

15.13 Ferramenta Proxy – tem como proposta mudar a base de requisição da URL. Sua Utilidade ocorre quando trabalhamos com um outro servidor como o Apache por exemplo.

```
import cherrypy
from cherrypy import tools

class Root:
    @cherrypy.expose
    def index(self):
        return "Base URL: %s %s " % (cherrypy.request.base,
                cherrypy.url(''))

    @cherrypy.expose
    def other(self):
        raise cherrypy.HTTPRedirect(cherrypy.url(''))

if __name__ == '__main__':
    conf = {'global': {'tools.proxy.on': True,
        'tools.proxy.base': 'http://someapp.net/blog',
        'tools.proxy.local': ''}}
    cherrypy.config.update(conf)
    cherrypy.quickstart(Root(), '/')
```

Digite http:localhost:8080/other e você será redirecionado para http://www.someapp.net/blog/

15.14 Ferramenta referênca – tem como objetivo filtrar as requisições baseadas em padrões. Requisições podem ser rejeitadas ou aceitas após encontrarmos um padrão.

```
import cherrypy
from cherrypy import tools
class Root:
    @cherrypy.expose
    def index(self):
        return cherrypy.request.headers.get('Referer')
if __name__ == '__main__':
    conf = {'/': {'tools.referer.on': True,
        'tools.referer.pattern': 'http://[^/]*dodgy\.com',
        'tools.referer.accept': False}}
    cherrypy.quickstart(Root(), '/', config=conf)
```

Todas requisições feitas do domínio e sub-domínio dodgy.com serão recusadas.

15.15 Ferramenta Resposta para cabeçalhos – O objetivo desta ferramenta é setar alguns ou todos cabeçalhos.

```
import cherrypy
```

```
from cherrypy import tools

class Root:
    @cherrypy.expose
    def index(self):
        return "Este Texto"

    @cherrypy.expose
    def outro(self):
        return "Outro texto"

if __name__ == '__main__':
    conf = {'/': {'tools.response_headers.on': True,
                  'tools.response_headers.headers': [('Content-Type',
                  'text/plain')]}}
    cherrypy.quickstart(Root(), '/', config=conf)
```

O conteúdo dos manipuladores de páginas foram setados de Content-Type para text/plain.

15.16 Ferramenta Trailing Slash – tem como objetivo fornecer uma maneira flexível de lidar com a barra final da requisição. Esta ferramenta está por default habilitada.

```
import cherrypy
from cherrypy import tools

class Root:
    @cherrypy.expose
    def index(self):
        return "This should have been redirected to add the trailing slash"


    @cherrypy.expose
    def nothing(self):
        return "This should have NOT been redirected"
    nothing._cp_config = {'tools.trailing_slash.on': False}



    @cherrypy.expose
    def extra(self):
        return "This should have been redirected to remove the trailing slash"
    extra._cp_config = {'tools.trailing_slash.on': True,
    'tools.trailing_slash.missing': False,
'   tools.trailing_slash.extra': True}

if __name__ == '__main__':
    cherrypy.quickstart(Root(), '/')
```


teste usando os seguintes comandos(chamadas URL:

http://localhost:8080	aceita comandos com barra
http://localhost:8080/nothing	desabilita os comandos com barra
http://localhost:8080/nothing/	desabilita os comandos com barra
http://localhost:8080/extra/	envia o comando para http://localhost:8080/extra

15.17 Ferramenta XML-RPC – tem como proposta transformar o CherryPy em um servidor XML-RPC e fazer manipuladores de páginas com chamadas XML-RPC.

```
import cherrypy
from cherrypy import _cptools

class Root:
    @cherrypy.expose
    def index(self):
        return "Regular web page handler"

class XMLRPCApp(_cptools.XMLRPCController):
    @cherrypy.expose
    def echo(self, message):
        return message

if __name__ == '__main__':
    root = Root()
    root.xmlrpc = XMLRPCApp()
    cherrypy.quickstart(root, '/')
```

Para testar digite no IDLE os seguintes comandos:

```
>>> import xmlrpclib
>>> s=xmlrpclib.ServerProxy('http://localhost:8080/xmlrpc')
>>> s.echo('test')
'test'
```

Não se esqueça de executar o CherryPy senão o sistema sinaliza erro. O erro abaixo foi conseguido executando o servidor CherryPy e depois de digitar os comandos o CherryPy foi desabilitado e tentamos executar o comando s.echo('test')

```
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    s.echo('test')
  File "C:\Python25\lib\xmlrpclib.py", line 1147, in __call__
    return self.__send(self.__name, args)
  File "C:\Python25\lib\xmlrpclib.py", line 1437, in __request
    verbose=self.__verbose
  File "C:\Python25\lib\xmlrpclib.py", line 1183, in request
    self.send_content(h, request_body)
  File "C:\Python25\lib\xmlrpclib.py", line 1297, in send_content
    connection.endheaders()
  File "C:\Python25\lib\httplib.py", line 860, in endheaders
    self._send_output()
```

```
  File "C:\Python25\lib\httplib.py", line 732, in _send_output
    self.send(msg)
  File "C:\Python25\lib\httplib.py", line 699, in send
    self.connect()
  File "C:\Python25\lib\httplib.py", line 683, in connect
    raise socket.error, msg
error: (10061, 'Connection refused')
```

15.18 Caixa de Ferramentas – todas as ferramentas CherryPy deve pertencer a uma caixa de ferramentas que deve ser gerenciada pelo motor CherryPy. Elas tem seu próprio espaço de nomes para evitar colisão de nomes. É possível ainda criar seus próprios nomes como segue:

```
from cherrypy._cptools import Toolbox,
mytb = Toolbox('mytb')
mytb.xml_parse = Tool('before_handler', xmlparse)
conf = {'/': {'mytb.xml_parse.on': True,
        'mytb.xml_parse.engine': 'amara'}}
```

15.19 Criando uma ferramenta – após decidirmos em nível ela irá atuar devemos utilizar um Hook para inserir a nova ferramenta. A construção da ferramenta é feita através do construtor de classe ou seja uma sub-classe de Tool da seguinte forma: Tool(point, callable, name=None, priority=50). Pondere sempre antes de criar ferramentas e use a referência bibliográfica para sanar dúvidas.

15.20 Servindo recursos estáticos – duas formas são fornecidas para este serviço: servir um arquivo ou servir um diretório. Veja exemplos:

Servindo arquivos:

```
#minhaapp.py
….
…..
…..
if __name__ == '__main__':
   import os.path
   current_dir = os.path.dirname(os.path.abspath(__file__))
   cherrypy.config.update({'environment': 'production',
                           'log.screen': True})
   conf = {'/': {'tools.staticfile.root': current_dir},
            '/css/style.css': {'tools.staticfile.on': True,
                           'tools.staticfile.filename':
                           'design1.css'}}
   cherrypy.quickstart(MinhaApp(), '/my', config=conf)
```

Neste exemplo definimos o diretório corrente para a aplicação e o diretório para o arquivo estático que necessariamente não precisa ser o mesmo. Note também que montamos a aplicação em /my
http://localhost:8080/my

```
Aplicação \
        minhaapp.py
        design1.css
```

Definindo um diretório completo para arquivos estáticos:

```
if __name__ == '__main__':
    import os.path
    current_dir = os.path.dirname(os.path.abspath(__file__))
    cherrypy.config.update({'environment': 'production',
                            'log.screen': True})
    conf = {'/': {'tools.staticdir.root': current_dir},
             '/static/css': {'tools.gzip.on': True,
                             'tools.gzip.mime_types':['text/css'],
                             'tools.staticdir.on': True,
                             'tools.staticdir.dir': 'data'},
            '/static/scripts': {'tools.gzip.on': True,
                             'tools.gzip.mime_types':
                             ['application/javascript'],
                             'tools.staticdir.on': True,
                             'tools.staticdir.dir': 'data'},
            '/feed': {'tools.staticdir.on': True,
                      'tools.staticdir.dir': 'feeds',
                      'tools.staticdir.content_types':
                       {'rss':'application/xml',
                           'atom': 'application/atom+xml'}}}
     cherrypy.quickstart(MyApp(), '/', config=conf)
```

Estrutura do diretório:

```
Aplicação /
        minhaapp.py
        data /
            design1.css
            some.js
        feeds /
            app.rss
            app.atom
```

Note que foi definido o caminho de arquivos CSS e JavaScript e que nós definimos o caminho para os recursos baseado na extensão do arquivo (.rss e .atom). Veja também foi mixado a localização do arquivo estático com o gzip com isto o arquivo será comprimido antes de ser enviado.

15.21 Bypassando a ferramenta estática para servidor de conteúdo estático – algumas vezes você pode querer reusar as funcionalidades internas do CherryPy sem usar as ferramentas estáticas diretamente. Isto é possível chamando a função serve_file de dentro de seu manipulador de página. Utilize a referência bibliográfica para ver exemplos.

16. Suporte WSGI – Web Server Gateway Interface (WSGI) está definida em PEP-333(Python Enhacement Proposal) e fornece acoplagem entre o servidor web e a aplicação web. Composta por três componentes Servidor ou Gateway, Middleware (módulo intermediário) e Aplicação ou Framework, o objetivo da WSGI é permitir que componentes sejam conectados com o mínimo de sobrecarga possível. Isto permite o reuso de funcionalidades como sessão, autenticação, envio de URL, logging etc... A partir da versão 3.0 foi implementada por Christian Wyglendowski este suporte. Salientamos que ferramentas CherryPy e módulos WSGI são diferentes em projeto mas não

em capacidade. Eles provém as mesmas funcionalidades de formas distintas e ferramentas CherryPy e módulos intermediários WSGI podem coexistir numa mesma aplicação.

Uma Servidor de aplicação WSGI dentro do Servidor CherryPy WSGI

```
import cherrypy
from paste.translogger import TransLogger

def application(environ, start_response):
    status = '200 OK'
    response_headers = [('Content-type', 'text/plain')]
    start_response(status, response_headers)
    return ['Hello world!\n']

if __name__ == '__main__':
    cherrypy.tree.graft(TransLogger(application), script_name='/')
    cherrypy.server.quickstart()
    cherrypy.engine.start()
```

Desta forma foi construído um servidor WSGI, que pode manipular aplicação WSGI sem qualquer problema. Com um detalhe, entretanto ferramentas e configurações não serão usadas em aplicação WSGI.

Aplicação tradicional em CherryPy com um Servidor WSGI:

```
import cherrypy
from cherrypy import tools
from wsgiref.simple_server import make_server
from flup.middleware.gzip import GzipMiddleware

class Root:
    @cherrypy.expose
    @tools.response_headers(headers=[('Content-Language', 'en-GB')])
    def index(self):
        return "Hello world!"

if __name__ == '__main__':
    wsgi_app = cherrypy.Application(Root(), script_name="/")
    cherrypy.engine.start(blocking=False)
    httpd = make_server('localhost', 8080, GzipMiddleware(wsgi_app))
    print "HTTP Serving HTTP on http://localhost:8080/"
    httpd.serve_forever()
```

17. Desenvolvendo um Site em CherryPy – neste exemplo mostrado a seguir temos um site desenvolvido em Python usando como servidor Web o CherryPY. Em primeiro lugar temos a página principal do site com fotografias, textos, link para cadastro de pedidos, link para tabela de preços, mensagens postadas pelo usuário e envio de mensagens para o site. Na página de cadastro de pedidos temos os campos a serem preenchidos e que depois serão enviados ao site que armazenará em um arquivo. Na página de tabela de preços temos o acesso ao banco de dados Mysql e visualização desta informação na tela do sistema. Este sistema possui 3 arquivos py , 1 css e outros

de imagem jpg: cabecalho.py, imagem2.py, pedido.py e default.css. Os arquivos de imagem não estão aqui listados, use arquivos do tipo jpg(img06.jpg, img05.jpg,.....).

#imagem2.py arquivo contendo o módulo principal com o cherrypy

```
# -*- coding: iso-8859-1 -*-   #alfabeto latino
import cherrypy
import os
import cabecalho
import pedido
class HelloWorld:

    def index(self,code=None,resposta=""):
        if code is None:
          code=""
        if True:
# permite ao cliente enviar mensagens
          recado_file=os.path.join(current_dir,'recado.txt')
          f = open(recado_file,'a')
          f.write(code)
          f.write("\n----------Mais Mensagens----------\n")
          f.close()
          resposta_file=os.path.join(current_dir,'mensagens.txt')
          f=open(resposta_file,'r')
          i=1
          for line in f.read():
             i=i+1
             if i==27:
               resposta+=line+"\n"
               i=1
             else:
               resposta+=line
          f.close()

        return cabecalho.cabecalho+cabecalho.esquerda+cabecalho.conteudo+\
              cabecalho.direita+resposta+cabecalho.direita1+cabecalho.rodape
    index.exposed = True

#caso a página não exista emite esta informação
    def default(self,qqcoisa):
        return "Página Inexistente"
    default.exposed=True

#cadastra pedido no arquivo de pedidos
                                                                                    def
cadastrapedido(self,nome="",endereco="",cidade="",estado="",cpf=None,email="",cpedido=""):
        if cpf!=None:
           if len(cpf)==0:
             return(pedido.erro)
        else:cpf=""
```

```
      pedido_file=os.path.join(current_dir,'pedido.txt')
      f=open(pedido_file,'a')
      f.write(nome+" "+endereco+" "+cidade+" "+estado+" "+cpf+" "+email+"\n")
      f.write(cpedido+"\n")
      f.close()

      return pedido.pedido
    cadastrapedido.exposed=True

#abre a tabela com preços e mostra ela na tela, usa banco de dados mysql
    def tabelapreco(self):
       import MySQLdb
       db=MySQLdb.connect("localhost","root","senha","testedb")
       cursor=db.cursor()
       sql="select * from preco"
       cursor.execute(sql)
       saida='''
       <meta http-equiv="content-type" content="text/html; charset=unicode" />
       <title>Argos Moveis Hospitalares</title>
       <meta name="keywords" content="móveis, hospital, cama, banqueta" />
       <meta name="Premium Series" content="" />
       <link href="default.css" rel="stylesheet" type="text/css" media="screen" />
       </head>
       <body>
       <!-- start header -->
       <div id="header">
          <div id="logo">
                  <h1><a
href="#"><span>Argos</span>Moveis<span>Hospitalares</span></a></h1>
                  <p>Fábrica de Móveis Hospitalares</p>
          </div>
          <div id="menu">
                  <ul id="main">
                          <li class="current_page_item"><a href="/">Home</a></li>

                  </ul>



          </div>
          <p>
          <p><h1>Tabela de Preços</h1></p>


       </div>
       '''
       saida1='<h3><p></p><table width="300" border="2" align="left">'
       for row in cursor.fetchall():
        saida1=saida1+"<tr>"
        saida1=saida1+"<td>%s</td>"  % row[0]
```

```
        saida1=saida1+"<td>%s</td>"  % row[1]
        saida1=saida1+"<td>%f</td>"  % row[2]
        saida1=saida1+"</tr>"
       saida2="</table><h3></body></html>"
       cursor.close()
       return saida+saida1+saida2
   tabelapreco.exposed=True

# envia um recado e grava em recado.txt a informacao
   def enviaarquivo(self):
       c=os.path.join(current_dir,'recado.txt')
       f=open (c,'rb')
       dados=f.read()
       f.close()
       return pedido.envia
   enviaarquivo.exposed=True

if(__name__=='__main__'):
   current_dir = os.path.dirname(os.path.abspath(__file__))
   conf = {
   '/': {'tools.staticfile.root': current_dir,'tools.encode.encoding':'ISO-8859-1',
   'tools.encode.on':True},
   'cadastrapedido': {'tools.staticfile.root': current_dir,'tools.encode.encoding':'ISO-8859-1',
   'tools.encode.on':True},
   '/default.css': {'tools.staticfile.on': True,
   'tools.staticfile.filename':'default.css'},
   '/img06.jpg': {'tools.staticfile.on':True,
   'tools.staticfile.filename':'img06.jpg'},
   '/img05.jpg': {'tools.staticfile.on':True,
   'tools.staticfile.filename':'img05.jpg'},
   '/img04.jpg': {'tools.staticfile.on':True,
   'tools.staticfile.filename':'img04.jpg'},
   '/img03.jpg': {'tools.staticfile.on':True,
   'tools.staticfile.filename':'img03.jpg'},
   '/img02.jpg': {'tools.staticfile.on':True,
   'tools.staticfile.filename':'img02.jpg'},
   '/img01.jpg': {'tools.staticfile.on':True,
   'tools.staticfile.filename':'img01.jpg'},
   '/vistaareafabrica.jpg': {'tools.staticfile.on':True,
   'tools.staticfile.filename':'vistaareafabrica.jpg'}

   }
   hello=HelloWorld()
   cherrypy.quickstart(hello,'/', config=conf)
```

#arquivo cabecalho.py

```
# -*- coding: iso-8859-1 -*-   #alfabeto latino
global recado,resposta
```

Python

```
recado=""
resposta="0i"
cabecalho="'
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<!--
Design by Free CSS Templates
http://www.freecsstemplates.org
Released for free under a Creative Commons Attribution 2.5 License

Name      : Premium Series
Description: A three-column, fixed-width blog design.
Version    : 1.0
Released   : 20090303

-->
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="content-type" content="text/html; charset=unicode" />
<title>Argos Moveis Hospitalares</title>
<meta name="keywords" content="móveis, hospital, cama, banqueta" />
<meta name="Premium Series" content="" />
<link href="default.css" rel="stylesheet" type="text/css" media="screen" />
</head>
<body>
<!-- start header -->
<div id="header">
        <div id="logo">
                <h1><a
href="#"><span>Argos</span>Moveis<span>Hospitalares</span></a></h1>
                <p>Fábrica de Móveis Hospitalares</p>
        </div>
        <div id="menu">
                <ul id="main">
                        <li class="current_page_item"><a href="#">Produtos</a></li>
                        <li><a href="#">Serviços</a></li>
                        <li><a href="#">Quem Somos</a></li>
                        <li><a href="#">Contate-nos</a></li>
                        <
                </ul>
                <ul id="feed">


                </ul>
        </div>

</div>
<!-- end header -->'"
esquerda="'
<div id="wrapper">
```

Python

```
        <!-- start page -->
        <div id="page">
                <div id="sidebar1" class="sidebar">
                        <ul>
                                <li>

                                        <h2>Faça seu pedido</h2>
                                        <ul>
                                                <li><a href="/cadastrapedido">Pedido Pessoa
Física.</a></li>
                                                <li><a href="#">Prudentina uma parceria que deu
certo.</a></li>

                                        </ul>
                                </li>
                                <li>
                                        <h2>Comentários Recentes</h2>
                                        <ul>
                                                <li><a href="#">Free CSS Templates</a> on <a
href="#">Melhoria dos Móveis</a></li>

                                        </ul>
                                </li>
                                <li>
                                        <h2>Categorias</h2>
                                        <ul>
                                                <li><a href="#">Aliquam libero</a></li>
                                                <li><a href="#">Consectetuer adipiscing elit</a></li>
                                                <li><a href="#">Metus aliquam
pellentesque</a></li>
                                                <li><a href="#">Suspendisse iaculis mauris</a></li>
                                                <li><a href="#">Urnanet non molestie
semper</a></li>
                                                <li><a href="#">Proin gravida orci porttitor</a></li>
                                        </ul>
                                </li>
                                <li>
                                        <h2>Tabela de Preços</h2>
                                        <ul>
                                                <li><a href="/tabelapreco">Móveis</a></li>

                                        </ul>
                                </li>
                        </ul>
                </div>'"

conteudo='"
                <!-- start content -->
                <div id="content">
                        <div class="flower"><img src="img06.jpg" alt="" width="510"
```

```
height="200" /></div>
                    <div class="post">
                        <h1 class="title"><a href="/index">Benvindo, aqui floresce uma parceria.</a></h1>
                        <p class="byline"><small>Postado em 11 de Novembro de 2009. by <a href="#">Antonio Sérgio Nogueira</a></small></p>
                        <div class="entry">
                            <p>A parceria <strong>Prudentina</strong> com a <strong>Argos</strong> tem como resultado a melhoria dos móveis hospitalares já produzidos e a redução dos prazos de entrega. Venha brindar nossa união com o seus pedidos, você vai sair satisfeito.<a href="http://www.capotasprudentina.com.br/"> Nossa fábrica.</a></p>
                        </div>
                    </div>
                    <div class="post">
                        <h2 class="title"><a href="#">Nossos Móveis</a></h2>
                        <p class="byline"><small>Postado em 11 de Novembro 2009. by <a href="#">Antonio Sérgio Nogueira</a></small></p>
                        <div class="entry">
                            <h3>Cama Hospitalar:</h3>
                            <blockquote>
                                <p> <img src="img05.jpg" alt="" width="250" height="125"/>
                            </blockquote>
                            <h3>Armário:</h3>
                            <ul>
                                <p> <img src="img04.jpg" alt="" width="250" height="125"/>
                            </ul>
                            <h3>Escada:</h3>
                            <ol>
                         <p> <img src="img03.jpg" alt="" width="250" height="125"/>

                            </ol>
                            <p class="links"><a href="#" class="more">&laquo;&laquo;  Mais: Veja Catálogo  &raquo;&raquo;</a></p>
                        </div>
                    </div>
                    <div class="post">
                        <h2 class="title"><a href="#">É Muito bom ter você como cliente. </a></h2>
                        <p class="byline"><small>Postado novembro, 2009 by <a href="#">Antonio Sérgio Nogueira</a></small></p>
                        <div class="entry">
                            <p>O clinte é nosso maior tesouro. Ligue-nos (18) 2104-1113. </p>
                            <p class="links"><a href="#" class="more">&laquo;&laquo;  Topo.  &raquo;&raquo;</a></p>
                        </div>
                    </div>
```

```
                </div>
                <!-- end content -->'"
direita="'

                <!-- start sidebars -->
                <div id="sidebar2" class="sidebar">
                        <ul>
                                <li>
                                        <form method="post" action="index">
                                                <div>
                                                        <h2>Deixe seu recado</h2>
                                                        <textarea name="code" rows=5
cols=25></textarea>
                                                        <input type="submit" value="Envia
Mensagem" />
                                                </div>
                                        </form>
                                </li>
                                <li>
                                          <h2>Comentários Postados</h2>'"
direita1="'

                                     </form>
                                </li>
                        </ul>
                </div>
                <!-- end sidebars -->
                <div style="clear: both;"> </div>
        </div>
        <!-- end page -->
</div>'"

rodape="'
<div id="footer">
        <p class="copyright">&copy;  2009 All Rights Reserved
 &bull;  Design by <a href="http://www.freecsstemplates.org/">Free CSS
Templates</a>.</p>
        <p class="link"><a href="#">Privacy Policy</a> &#8226; <a
href="#">Terms of Use</a></p>
</div>
</body>
</html>

'"
```

 # arquivo pedido.py

```
# -*- coding: iso-8859-1 -*-
```

Python

```
envia='''
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=unicode" />
<title>Argos Moveis Hospitalares</title>
<meta name="keywords" content="móveis, hospital, cama, banqueta" />
<meta name="Premium Series" content="" />
<link href="default.css" rel="stylesheet" type="text/css" media="screen" />
</head>
<body>
<!-- start header -->
<div id="header">
	<div id="logo">
		<h1><a
href="#"><span>Argos</span>Moveis<span>Hospitalares</span></a></h1>
		<p>Fábrica de Móveis Hospitalares</p>
	</div>
	<div id="menu">
		<ul id="main">
			<li class="current_page_item"><a href="/">Home</a></li>

		</ul>



	</div>
	<p>
	<p><h1>Catálogo Enviado</h1></p>

</div>
<!-- end header -->
</body>
</html>
'''

erro='''
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<!--
Design by Free CSS Templates
http://www.freecsstemplates.org
Released for free under a Creative Commons Attribution 2.5 License

Name	: Premium Series
Description: A three-column, fixed-width blog design.
Version	: 1.0
Released	: 20090303

-->
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head>
<meta http-equiv="content-type" content="text/html; charset=unicode" />
<title>Argos Moveis Hospitalares</title>
<meta name="keywords" content="móveis, hospital, cama, banqueta" />
<meta name="Premium Series" content="" />
<link href="default.css" rel="stylesheet" type="text/css" media="screen" />
</head>
<body>
<!-- start header -->
<div id="header">
	<div id="logo">
		<h1><a
href="#"><span>Argos</span>Moveis<span>Hospitalares</span></a></h1>
		<p>Fábrica de Móveis Hospitalares</p>
	</div>
	<div id="menu">
		<ul id="main">
			<li class="current_page_item"><a href="/cadastrapedido">Home</a></li>

		</ul>
		<ul id="feed">


		</ul>
	</div>

</div>
<!-- end header -->
<h1>Erro no preenchimento do pedido</h1>
</body>
</html>
'''



pedido= '''
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<!--
Design by Free CSS Templates
http://www.freecsstemplates.org
Released for free under a Creative Commons Attribution 2.5 License

Name      : Premium Series
Description: A three-column, fixed-width blog design.
Version    : 1.0
Released   : 20090303

-->
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
```

Python

```
<meta http-equiv="content-type" content="text/html; charset=unicode" />
<title>Argos Moveis Hospitalares</title>
<meta name="keywords" content="móveis, hospital, cama, banqueta" />
<meta name="Premium Series" content="" />
<link href="default.css" rel="stylesheet" type="text/css" media="screen" />
</head>
<body>
<!-- start header -->
<div id="header">
	<div id="logo">
		<h1><a
href="#"><span>Argos</span>Moveis<span>Hospitalares</span></a></h1>
		<p>Fábrica de Móveis Hospitalares</p>
	</div>
	<div id="menu">
		<ul id="main">
			<li class="current_page_item"><a href="/index">Home</a></li>

		</ul>
		<ul id="feed">


		</ul>
	</div>

</div>
<!-- end header -->

<div id="wrapper">
	<!-- start page -->
	<div id="page">
		<div id="sidebar1" class="sidebar">
			<ul>
				<li>

					<h2>Produtos + vendidos</h2>
					<ul>
						<li><img	src="img05.jpg"	alt=""	width="200"
height="80" />
				<h3> Cod:001-Cama Hospitalar</h3>
						</li>
						<li><img	src="img04.jpg"	alt=""	width="200"
height="80" />
				<h3> Cod:002-Armário 1 porta</h3>
						</li>
						<li><img	src="img03.jpg"	alt=""	width="200"
height="80" />
				<h3> Cod:003-Escada Hospitalar</h3>
						</li>
```

```
                                </ul>
                        </li>

                        <li>
                                <h2>Catálogos</h2>
                                <ul>
                                        <li><a href="#">Camas</li>
                                        <li><a href="#">Cadeiras</a></li>
                                        <li><a href="#">Suportes</a></li>
                                        <li><a href="#">Produtos Prudentina</a></li>
                                </ul>
                        </li>
                </ul>
        </div>


        <!-- start content -->
        <div id="content">
                <div class="flower"><img src="vistaareafabrica.jpg" alt="" width="510" height="265" /></div>
                <div class="post">
                        <h1 class="title"><a href="/index">Benvindo, é aqui que fabricamos seus móveis.</a></h1>
                        <p class="byline"><small>Postado em 11 de Novembro de 2009. by <a href="#">Antonio Sérgio Nogueira</a></small></p>
                        <div class="title" form method="get" action="/">

                                <h2>Cadastro Pessoa Física</h2>
                                <form action="" method="post">
                                <h3>*Nome-----: <input name="nome" size=40 />
                    <p> *Endereço:<input name="endereco" size=40 />
                    <p> *Cidade---:<input name="cidade" size=30/>
                    *Estado:<input name="estado" size=2/>
                    <p> *CPF-------:<input name="cpf" size=11/>
                    <p> *Email----:<input name="email" size=100/>
                    <p><h2>Faça seu pedido aqui</h2>
                    <p><h3>Colocando:<h3>
                    <p><h3>Quantidade|Codigo|Descricao</h3>
                    <p><textarea name="cpedido" rows=30 cols=50> </textarea>

                    <p>   <input type="submit" value="Enviar"
                                </form></h3>
                                <p class="links"><a href="#" class="more">&laquo;&laquo;  Mais: Veja Catálogo  &raquo;&raquo;</a></p>
                        </div>
                </div>
                <div class="post">
                        <h2 class="title"><a href="#">É Muito bom ter você como cliente.
```

```
</a></h2>
                        <p class="byline"><small>Postado novembro, 2009 by <a href="#">Antonio Sérgio Nogueira</a></small></p>
                        <div class="entry">
                            <p>O clinte é nosso maior tesouro. Ligue-nos (18) 2104-1113. </p>
                            <p class="links"><a href="#" class="more">&laquo;&laquo;  Topo.  &raquo;&raquo;</a></p>
                        </div>
                    </div>
                </div>
                <!-- end content -->
</body>
</html>'"
```

#arquivo default.css

```
/*
Design by Free CSS Templates
http://www.freecsstemplates.org
Released for free under a Creative Commons Attribution 2.5 License
*/

body {
    margin-top: 10px;
    padding: 0;
    background: #787878;
    text-align: justify;
    font-family: Georgia, "Times New Roman", Times, serif;
    font-size: 12px;
    color: #616161;
}

h1, h2, h3 {
    margin-top: 0;
    color: #8C0209;
}

h1 {
    font-size: 1.6em;
    font-weight: normal;
}

h2 {
    font-size: 1.6em;
}

h3 {
    font-size: 1em;
```

```
      color: black;
}

ul {
}

a {
        text-decoration: none;
        color: #8C0209;
}

a:hover {
        border-bottom: none;
}

a img {
        border: none;
}

img.left {
        float: left;
        margin: 0 20px 0 0;
}

img.right {
        float: right;
        margin: 0 0 0 20px;
}

#header {
        width: 1000px;
        margin: 0 auto;
        height: 150px;
        background: url(images/img02.jpg) repeat-x left top;
}

/* Header */

#logo {
        width: 1000px;
        height: 60px;
        margin: 0 auto;
        padding: 0 10px 0 70px;
        background: url(images/img01.jpg) no-repeat left top;
}

#logo h1, #logo p {
        float: left;
        margin: 0;
        color: #8C0209;
```

```
}

#logo span {
	color: #000000;
}

#logo h1 {
	padding: 25px 0 0 0;
	letter-spacing: -1px;
	text-transform: lowercase;
	font-weight: normal;
	font-size: 3em;
}

#logo p {
	text-transform: uppercase;
	padding: 47px 0 0 3px;
	font-size: 10px;
	color: #110E0F;
}

#logo a {
	border: none;
	text-decoration: none;
	color: #8C0209;
}

/* Menu */

#menu {
	width: 1000px;
	margin: 0 auto;
	padding: 0;
	height: 60px;
	background: url(images/img02.jpg) no-repeat left top;
}

#menu ul {
	margin: 0;
	padding: 0;
	list-style: none;
}

#menu li {
	display: inline;
}

#menu a {
	display: block;
	float: left;
```

```
        height: 32px;
        margin: 0;
        padding: 18px 30px 0 30px;
        text-decoration: none;
        text-transform: capitalize;
        background: url(images/img03.jpg) no-repeat right top;
        font-family: Georgia, "Times New Roman", Times, serif;
        font-size: 12px;
        color: #FFFFFF;
}

#menu a:hover {
        color: #FFFFFF;
}

#menu .current_page_item a {
        color: #FFFFFF;
}

/* Wrapper */

#wrapper {
}

/* Page */

#page {
        width: 990px;
        margin: 0 auto;
        padding: 20px 5px;
        background: #FFFFFF;
}

#page-bg {
}

/* Latest Post */

#latest-post {
        padding: 20px;
        border: 1px solid #E7E7E7;
}

/* Content */

#content {
        float: left;
        width: 550px;
}
```

Python

```
.post {
        padding-bottom: 15px;
        line-height: 200%;
}

.post h1 {
        font-weight: normal;
}

.title {
        margin: 0;
        padding: 10px 0 4px 20px;
        font-weight: normal;
}

.title a {
        border-bottom: none;
        color: #8C0209;
}

.title a:hover {
        border-bottom: 1px dotted #000000;
}

.byline {
        border-bottom: 1px #BBBBBB dashed;
        margin: -10px 20px 20px 20px;
}

.tag {
        padding: 0 15px;
}

.entry {
        padding: 0 20px;
}

.links {
        padding: 4px 0px;
        text-align: right;
        font-weight: bold;
}

.links a {
        border: none;
}

.links a:hover {
}
```

```
/* Sidebars */

#sidebar1 {
	float: left;
}

#sidebar2 {
	float: right;
}

.sidebar {
	float: left;
	width: 220px;
	padding: 0;
	font-size: 12px;
}

.sidebar ul {
	margin: 0;
	padding: 0;
	list-style: none;
}

.sidebar li {
	padding: 0 0 20px 0;
}

.sidebar li ul {
}

.sidebar li li {
	margin: 0 20px 0 15px;
	padding: 8px 0px;
	border-bottom: 1px #BBBBBB dashed;
}


.sidebar li h2 {
	height: 30px;
	margin: 0 0 0 0;
	padding: 10px 15px 0px 15px;
	background: #890208 url(images/img05.jpg) no-repeat left top;
	letter-spacing: -1px;
	font-size: 16px;
	color: #FFFFFF;
}

.sidebar a {
}
```

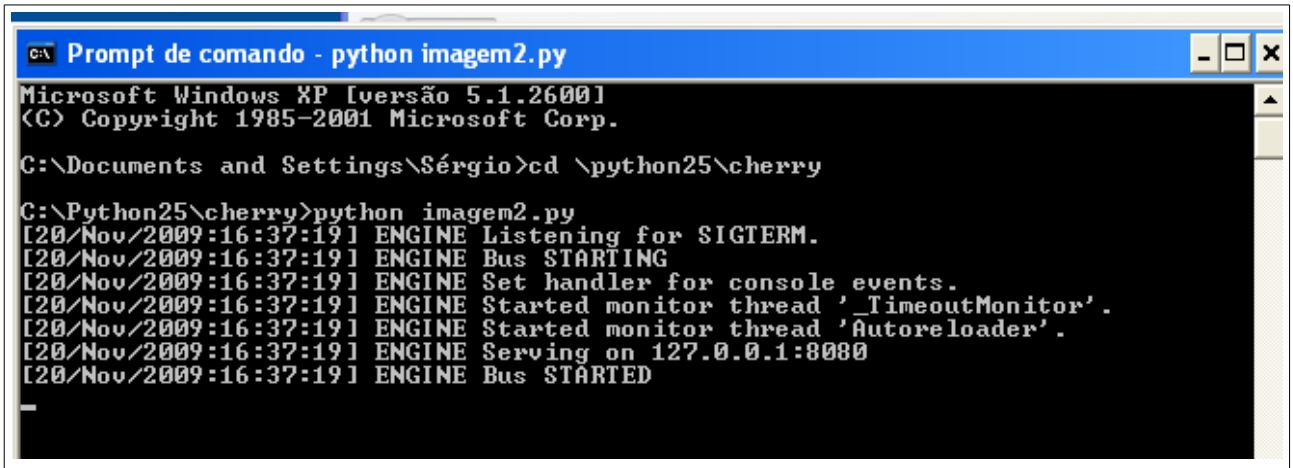
```
/* Search */

#searchform {
	margin: 0;
	padding: 0 0 0 0;
}

#searchform br {
	display: none;
}

#searchform h2 {
}

#s {
	margin: 10px 0px 0 15px;
	padding: 2px 2px;
	width: 180px;
	height: 18px;
	border: 1px solid #CA8186;
	background: #FFFFFF;
	font-size: 10px;
	color: #000000;
}

#x {
	margin: 0;
	padding: 2px 5px;
	height: 25px;
	background: #CA8186;
	text-decoration: none;
	text-transform: uppercase;
	font-family: Arial, Helvetica, sans-serif;
	font-size: 10px;
	color: #CCCCCC;
}
/* Calendar */

#calendar_wrap {
	padding: 0 15px;
	text-align: center;
}

#calendar_wrap table {
	width: 100%;
}

#calendar_wrap th {
}
```

```
#calendar_wrap td {
}

#calendar_wrap tfoot td {
	border: none;
}

#calendar_wrap tfoot td#prev {
	text-align: left;
	font-weight: bold;
	border: none;
}

#calendar_wrap tfoot td#prev a {
	border: none;
}

#calendar_wrap tfoot td#next {
	text-align: right;
	font-weight: bold;
	border: none;
}

#calendar_wrap tfoot td#next a {
	border: none;
}

/* Footer */

#footer {
	width: 960px;
	height: 70px;
	margin: 0 auto;
	padding: 0 20px;
	background: url(images/img04.jpg) no-repeat left top;
}

#footer p {
	margin: 0;
	padding: 25px 0 0 0;
	text-align: center;
	font-size: smaller;
}

#footer a {
}

#footer .link {
	float: right;
}
```

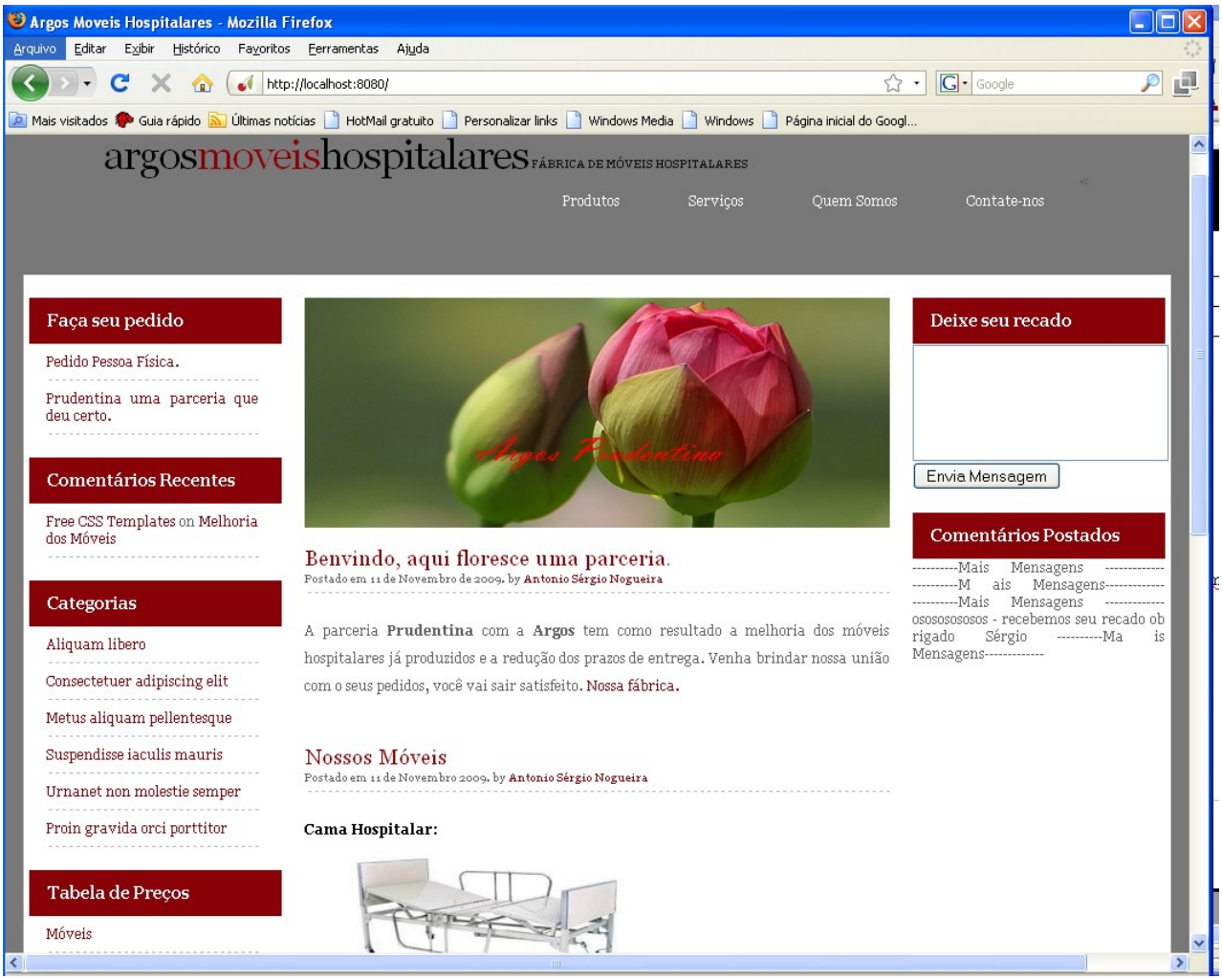
Python

```
#footer .copyright {
	float: left;
}

.flower {
	padding-left: 20px;
}
```

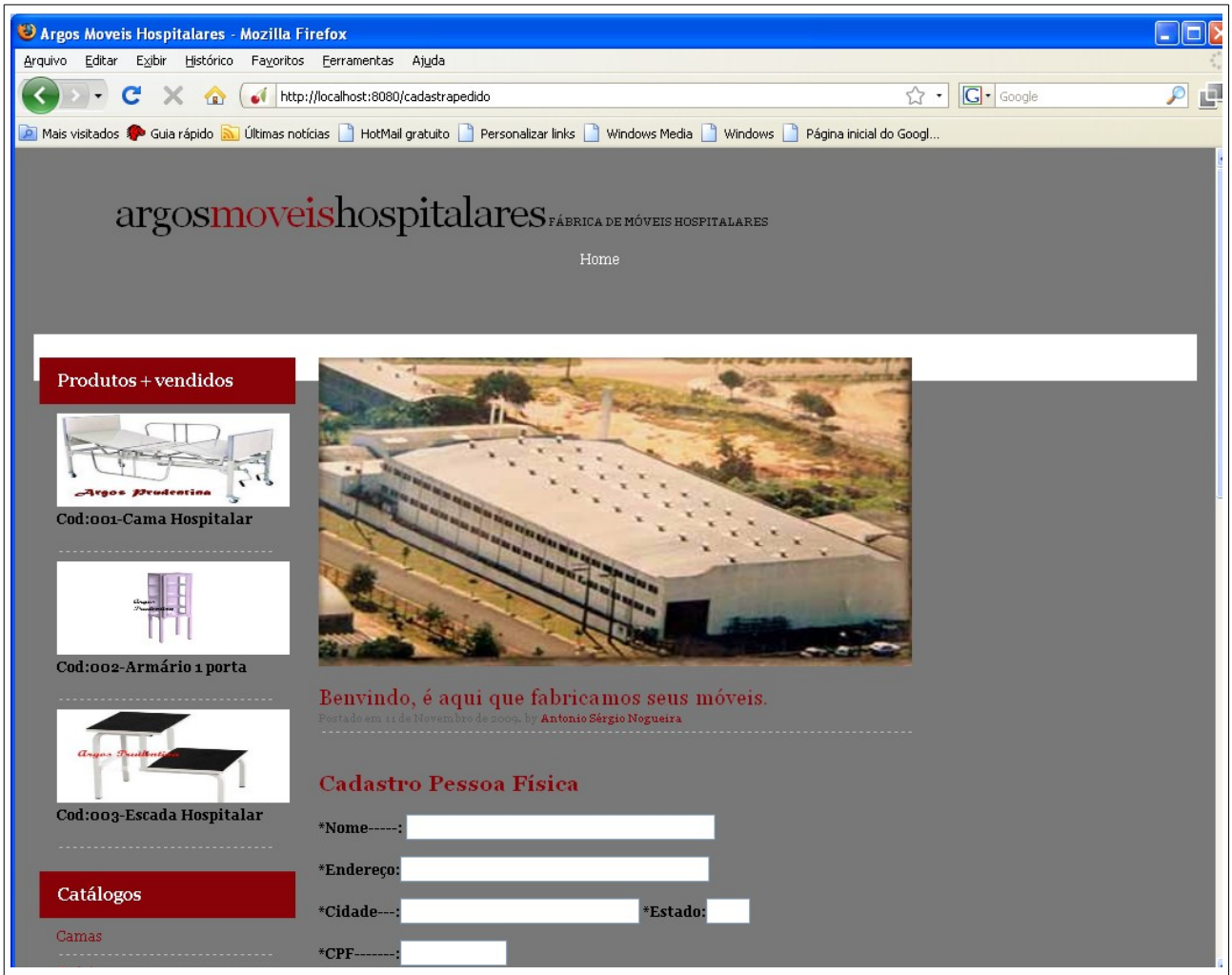
Nosso Site:



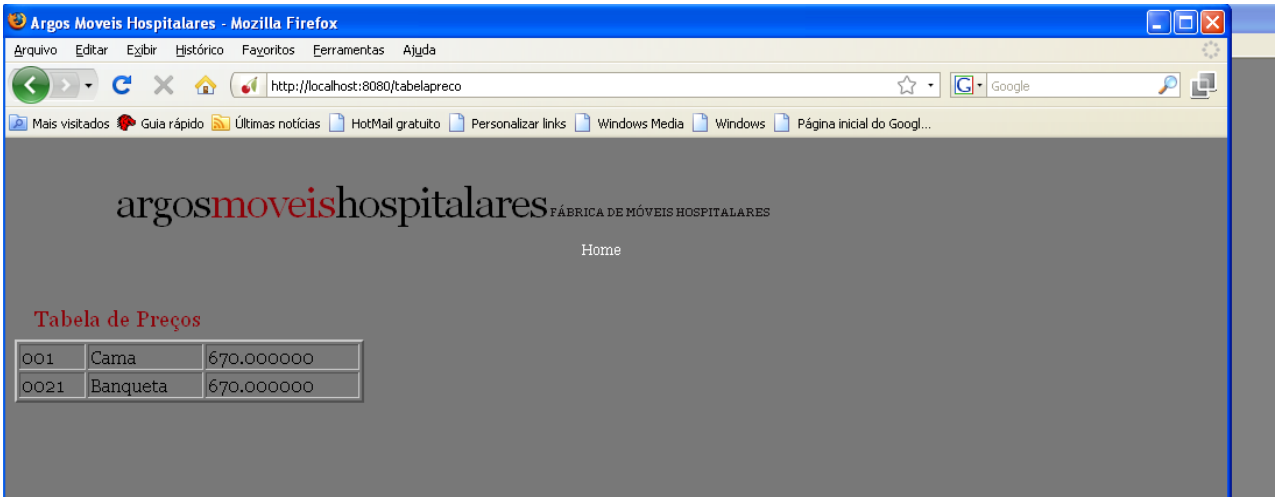
Rodando o servidor web CherryPY



Página principal do site



Cadastro de Pedidos



Acesso a tabela de preços com o Mysql.

Referência Bibliográfica e pacotes:

Python

http://pypi.python.org/packages/2.5/s/setuptools/setuptools-0.6c11.win32-py2.5.exe

http://pypi.python.org/packages/2.6/R/Routes/Routes-1.11-py2.6.egg

http://pypi.python.org/packages/2.6/R/Routes/Routes-1.11-py2.6

- Hellegouarch Sylvan, Cherry Essentials – Rapid Python Web Application Development, Ed.: Packt Publishing – 2007

http://www.cherrypy.org/wiki/CherryPyTutorial